

# Quantum Distributed Network Computing: Lower Bounds and Techniques

Michael Elkin\*    Hartmut Klauck†    Danupon Nanongkai‡    Gopal Pandurangan§

## Abstract

We study lower bounds for quantum distributed network computing, where a set of nodes (representing quantum computers) interconnected by an underlying network consisting of (bandwidth-restricted) quantum communication links, communicate using quantum communication. We show non-trivial time lower bounds of quantum distributed algorithms for fundamental graph problems. Our bounds are strong in the sense that they hold even when nodes have unlimited computational power and share an unlimited number of entangled qubits. Our bounds hold for many fundamental graph *verification* problems as well as for various graph *optimization* problems (both exact and approximate optimization). Our lower bounds are shown in a uniform way by showing a connection between quantum communication complexity and quantum distributed computing and hence gives a general technique for showing lower bounds for a wide variety of distributed network problems in the quantum setting.

For many graph verification problems, we show a *tight* two-sided error quantum lower bound of  $\tilde{\Omega}(\sqrt{n})$  time, where  $n$  is number of nodes in the network ( $\tilde{\Theta}(x)$  hides  $\text{poly log } x$ ). Our lower bounds match the previous classical deterministic algorithms and even subsume the previously known lower bounds in the classical distributed computing model [Das Sarma et al., STOC 11, Peleg and Rubinfeld, FOCS 99, Elkin STOC 04]. In particular, our quantum lower bounds for the *Hamiltonian cycle* and *spanning tree verification* problems imply a new bound also in the classical setting which answers an open problem in [Das Sarma et al., STOC 11]. They are also the *first randomized lower bounds* for both graph problems, subsuming the previous deterministic lower bounds [Das Sarma et al., STOC 11]. We then show a  $\tilde{\Omega}(\min(W/\alpha, \sqrt{n}))$ -time lower bound ( $W$  is the ratio between the maximum and minimum edge weights in the graph) for any  $\alpha$ -approximation quantum distributed algorithm for many graph optimization (both exact and approximate) problems such as *minimum spanning tree*, *shortest path tree*, *minimum cut* etc. In particular, our bound for the classical minimum spanning tree problem, besides generalizing the bound in [Das Sarma et al., STOC 11] to the quantum setting, also improves the previous bound even in the classical setting, and is the first bound that is *tight for all values of  $W$*  as it matches previous deterministic upper bounds [Elkin, STOC 04].

To prove our quantum lower bounds, we introduce a new communication model called the *server model* which seems to be a key model that captures the hardness of quantum distributed networks. This model is a generalization of the standard two-party (Yao's communication complexity) model and lower bounds on this model imply lower bounds on the two-party model and quantum distributed network model. We prove quantum lower bounds in the server model, some of which also imply *new* lower bounds in the two-party model for problems such as Hamiltonian cycle and spanning tree, even in the classical setting. We extend our study to the *gap version* of the graph verification problems, which is a key to get tight lower bounds for all values of  $W$ . We show a tight one-sided error lower bound of the *gap-Hamiltonian cycle*, *gap-spanning tree*, and *gap-connectivity* problem which might be of independent interest.

---

\*Department of Computer Science, Ben-Gurion University, Beer-Sheva, 84105, Israel. E-mail: [elkinm@cs.bgu.ac.il](mailto:elkinm@cs.bgu.ac.il).

†Division of Mathematical Sciences, Nanyang Technological University, Singapore 637371 & Centre for Quantum Technologies, National University of Singapore, Singapore 117543. E-mail: [hklauck@gmail.com](mailto:hklauck@gmail.com). Research at the Centre for Quantum Technologies is funded by the Singapore Ministry of Education and the National Research Foundation.

‡Theory and Applications of Algorithms Research Group, University of Vienna, 1090-Vienna, Austria. E-mail: [danupon@gmail.com](mailto:danupon@gmail.com).

§Division of Mathematical Sciences, Nanyang Technological University, Singapore 637371 & Department of Computer Science, Brown University, Providence, RI 02912, USA. E-mail: [gopalpandurangan@gmail.com](mailto:gopalpandurangan@gmail.com). Supported in part by the following research grants: Nanyang Technological University grant M58110000, Singapore MOE Academic Research Fund (AcRF) Tier 2 grant MOE2010-T2-2-082, and a grant from the US-Israeli Binational Science Foundation (BSF).

## 1 Introduction

This paper concerns the marriage of two major computer science areas: distributed network computing and quantum computing. The power and limitations of distributed/decentralized network computation has been studied intensively over the last three decades or so. In a distributed network, each individual node can directly communicate only with its neighboring nodes. Despite being restricted to such *local* communication, the network itself should work towards a *global* task, e.g., broadcast a piece of data, elect a (global) leader, construct a spanning tree or a minimum spanning tree (MST), construct a shortest path tree (SPT) or efficient routing paths between node pairs. Indeed, such fundamental primitives underlie the operation of modern communication networks.

Quantum computing, on the other hand holds the promise to fundamentally change the way we do computing. Quantum computing and distributed computing might enter a mutually beneficial partnership. On the one hand, it may be easier to build a number of small quantum computers and connect them distributively rather than build a single large one (which seems quite difficult under current technology). On the other hand, some of the fundamental problems in distributed computing could potentially be solved more efficiently by taking advantage of the superior resources and processing power that quantum mechanics offers. This paper focuses on this second aspect, namely studying the power of distributed network computation augmented with quantum effects.

Some distributed tasks can be solved entirely via local communication, for instance, how many friends of friends one has. Classic combinatorial optimization problems such as maximal matching, coloring, dominating set, vertex cover, or approximations thereof are considered “local” problems, as they can be shown to be solved using small (i.e., polylogarithmic) local communication (e.g., see [54, 38, 62]). However many important optimization problems (e.g., those listed in the above paragraph) are “global” problems from the distributed computation point of view. To count the total number of nodes, to elect a leader, to compute a spanning tree, a MST or a SPT, information necessarily must travel to the farthest nodes in a system. If exchanging a message over a single edge costs one time unit, one needs  $\Omega(D)$  time units to compute the result, where  $D$  is the network diameter. If message size was unbounded, one can simply collect all the information in  $O(D)$  time, and then compute the result. However, in many applications, there is *bandwidth restriction* on the size of the message (or the number of bits) that can be exchanged over a communication link in one time unit. This motivates studying global problems in the  $B$ -model (also commonly known as the CONGEST( $B$ ) model [54]), where each node can exchange at most  $B$  bits (typically  $B$  is small, say  $O(\log n)$ ) among its neighbors in one time step. This is one of the central models in the study of distributed computation. The design of efficient algorithms for the  $B$ -model (in the classical setting), as well as establishing lower bounds on the time complexity of various fundamental distributed computing problems, has been the subject of an active area of research called (locality-sensitive) *distributed computing* for the last three decades (see [54] and references therein). In particular, extensive research has been done in studying upper and lower bounds on distributed algorithms for various fundamental graph optimization and verification (both exact and approximate) problems (e.g., see [54, 48, 65, 20, 18, 45, 30, 31, 38, 62] and the work of [14] and the references therein).

The main focus of the current work is to understand the time complexity of fundamental global problems in the  $B$ -model in the *quantum setting*, where nodes can use quantum processing and can communicate over quantum links using quantum bits and can use exclusively quantum phenomena such as *entanglement* (e.g., see [17, 7, 26]). In a recent work ([14]) it was shown that many important graph verification and optimization problems in the  $B$ -model in the classical setting have a time lower bound of  $\tilde{\Omega}(\sqrt{n})$ , where  $n$  is the number of nodes in the distributed network. (Note that this bound holds even in a network of small ( $O(\log n)$ ) diameter.) The present work significantly extends and generalizes the above result and shows that the same time lower bound essentially holds for the quantum setting as well for various important graph verification and optimization problems. To

prove our main result, we develop new techniques that relate quantum communication complexity lower bounds to quantum distributed computing lower bounds. Our techniques are general and can be used to derive non-trivial lower bounds for a wide variety of distributed network problems in the quantum setting.

**Organization of the paper** In Section 2 we define the quantum distributed network model used in this paper. For the sake of readability, this is first done informally, while rigorous formal definitions are presented in Part II of the Appendix. In Section 3 we discuss the role of quantum effects in distributed computing, especially in relation to local and global problems, and mention other work that relates quantum and distributed computing. In Section 4 we state the graph problems considered in this paper. Section 5 states our main technical results — our lower bound theorems for graph verification and optimization problems as well as some additional new lower bounds results in quantum communication complexity. Full proofs for all our theorems are given in the Appendix (Parts I and II). Section 6 gives an overview of our proof ideas and techniques used. We conclude in Section 7 with some open problems.

## 2 Distributed Computing Model

Consider a synchronous network of processors modeled by an undirected  $n$ -vertex graph, where nodes model the processors and edges model the links between the processors. The processors (henceforth, nodes) communicate by exchanging messages via the links/channels (edges). The vertices have limited global knowledge, in particular, each of them has its own local perspective of the network (a.k.a. graph), which is confined to its immediate neighborhood. The nodes may have to compute (cooperatively) some global function of the graph, such as a spanning tree (ST) or a minimum spanning tree (MST), via communicating with each other and running a distributed algorithm designed for the task at hand.

There are several measures to analyze the performance of such algorithms, a fundamental one being the running time, defined as the worst-case number of *rounds* of distributed communication. This measure naturally gives rise to a complexity measure of problems, called the *time complexity*. We assume the  $B$ -model, mentioned above: In each round at most  $B$  bits can be sent through each edge in each direction, where  $B$  is the bandwidth parameter of the network.

In the standard distributed computing literature (e.g., see [54, 48, 65]), the usual model considered is the classical deterministic model where nodes (representing classical processors) communicate using classical bits and the algorithms used by nodes are deterministic. An additional resource in the classical setting that makes algorithms more powerful is *randomness*. In the *private randomness* model, each node has an access to an infinite random string. A seemingly more powerful model is the *shared randomness* model where nodes have access to an infinite shared random string.

In the quantum setting, a distributed network could be augmented with two additional resources: *quantum communication* and *entanglement* (see e.g., [17]). Quantum communication allows nodes to communicate with each other using quantum bits (qubits). Entanglement allows nodes to possess qubits that are entangled with qubits of other nodes. For example, a well-known entangled state on two qubits is the *EPR* pair [19, 4] which is a pair of qubits that, when measured, will either both be zero or both be one, with probability  $1/2$  each. An EPR pair shared by two nodes can hence be used to, among other things, generate a shared random bit for the two nodes. Assuming entanglement implies shared randomness (even among all nodes), but also allows for other operations such as quantum teleportation [51], which replaces quantum communication by classical communication plus entanglement. Quantum communication and entanglement resources are known to help saving communication in some cases, as will be illustrated later (cf. Example 3.1 in Section 3).

Quantum distributed networks can be categorized based on which resources are assumed. In this paper, we are interested in the most powerful model where both quantum communication and

entanglement are assumed. In a quantum distributed network with  $n$  nodes we allow the nodes to share an arbitrary  $n$ -partite entangled state that does not depend on the input. For instance, every pair of nodes could share some EPR-pairs, and all  $n$  nodes together could share instances of the superposition  $(1/\sqrt{2})(|0^n\rangle + |1^n\rangle)$  in order to simulate a global public coin. To put it differently, we allow the most general form of shared entanglement as long as it does not reveal anything about the problem inputs. Throughout the paper, we simply refer to this model as quantum distributed network (or just distributed network, if the context is clear). All lower bounds we show in this paper hold in this model, and thus imply lower bounds in weaker models as well.

### 3 The Role of Quantumness in Local and Global Problems

Lower bounds for local problems (e.g., maximal independent set [37]), where the running time is usually  $O(\text{poly log } n)$ , in the quantum setting usually directly follow from the same arguments as in the classical setting. This is because these lower bounds are proved using the “limited sight” argument: The nodes do not have time to get the information of the entire network. Since entanglement *cannot be used to replace communication* (by, e.g., Holevo’s theorem [27] (also see [51, 50])), the same argument holds in the quantum setting with prior entanglement. This argument is captured by the notion of *physical locality* defined by Gavioille et al. [26], where it is shown that for many *local* problems, quantumness does not give any significant speedup in time compared to the classical setting.

The above argument, however does not seem to be extendible to *global* problems where the running time is usually  $\Omega(D)$  where  $D$  is the diameter of the network. In this setting, the argument developed in the line of work in [14] (which follows the line of work in [55, 47, 22, 35]) can be used to show tight lower bounds of many problems in the classical setting. However, this argument does not always hold in the quantum setting. The main reason is that this argument essentially relies on the “congestion” argument: Nodes cannot communicate fast enough (due to limited bandwidth) to get important information to solve the problem. However, we know that the quantum communication and entanglement can potentially *decrease the amount of communication* and thus there might be some problems that can be solved faster. To illustrate this point, consider the following contrived example where we solve the *set disjointness problem*.

**Example 3.1.** Suppose we give  $b$ -bit string  $x$  and  $y$  to node  $u$  and  $v$  in the network, respectively, where  $b = \sqrt{n}$ . We want to check whether the inner product  $\langle x, y \rangle$  is zero or not. This is called the *Set Disjointness* problem (Disj). It is easy to show that it is impossible to solve this problem in less than  $D/2$  rounds since there will be no node having the information from both  $u$  and  $v$  if  $u$  and  $v$  are of distance  $D$  apart. This is the very basic idea of the limited sight argument. (We note that the real argument to prove natural problems are sometimes much more complicated than this, see, e.g., [37, 36, 38].) This argument holds for both classical and quantum setting and thus we have a lower bound of  $\Omega(D)$  on both settings. This lower bound, in fact, can be improved to  $\tilde{\Omega}(b) = \tilde{\Omega}(\sqrt{n})$  in the classical setting, even when the network has diameter  $O(\log n)$ . This follows from the communication complexity of  $\Omega(b)$  of Disj [2, 29, 3, 59] and the *Simulation Theorem* of [14]. This lower bound, however, does not hold in the quantum setting since we can simulate the known  $O(\sqrt{b})$ -communication quantum protocol ([1]) in  $O(\sqrt{b}D) = O((n)^{1/4}D)$  rounds.

Thus we have an example where quantum communication gives an advantage over classical communication. A fundamental question is: “Does this phenomenon occur for natural distributed network problems?” In this paper we show that, for many global graph problems, the lower bounds that hold for the classical setting also hold in the quantum setting. In particular, for the well-studied distributed MST problem, our results show that quantumness does not give any advantage compared to the classical setting.

**Other Related Work** While our work focuses on solving graph problems in quantum distributed networks, there are several prior works focusing on other distributing computing problems (including communication complexity in Yao’s two-party or multiparty communication model) using quantum effects. We note that fundamental distributed computing problems such as leader election and byzantine agreement have been shown to be solved better using quantum phenomena (see e.g., [17]). It has been shown that leader election in anonymous networks can be solved using quantum communication even without pre-shared entanglement [64, 17]; in the classical setting this is impossible. In [5], it has been shown that byzantine agreement against a dynamic adversary can be solved more efficiently using quantum communication. Entanglement has been used to reduce the amount of communication of a specific function of input data distributed among 3 parties [12] (see also the work of [9] on multiparty quantum communication complexity). Many related results of the above scenario are discussed in [15, 63]. As of now no separation between quantum and randomized communication complexity is known in the so-called number-on-the-forehead multiparty model, in which players’ inputs overlap.

There are several results showing that quantum communication complexity in the two-player model can be more efficient than classical randomized communication complexity, see for instance [10, 56]. These results also easily extend to the so-called number-in-hand multiparty model (in which players have separate inputs). Other papers relating quantum and distributed computing are [8], [11], [33], [34], [52], and [24].

## 4 Graph Problems

**Graph problems in the quantum distributed network model** Our main interest is on two types of graph problems: optimization and verification problems. In both types of problems, we are given a distributed network  $N$  modeled by a graph and some property  $\mathcal{P}$  such as “Hamiltonian cycle”, “spanning tree” or “connected component”.

In optimization problems, we are additionally given a (positive) weight function  $w : E(N) \rightarrow \mathbb{R}_+$ , and our goal is to find a subnetwork  $M$  of  $N$  of minimum weight that satisfies  $\mathcal{P}$  (e.g. minimum Hamiltonian cycle or minimum spanning tree) where every node knows which edges incident to it are in  $M$ . Algorithms sometimes depend on a parameter  $W$  defined as  $W = \frac{\max_{e \in E(N)} w(e)}{\min_{e \in E(N)} w(e)}$ .

In verification problems, we are additionally given a subnetwork  $M$  of  $N$  (each node knows which edges incident to it are in  $M$ ). We want to determine whether  $M$  has some property, e.g.,  $M$  is Hamiltonian cycle ( $\text{Ham}(N)$ ), a spanning tree ( $\text{ST}(N)$ ), or connected component ( $\text{Conn}(N)$ ).  $M$  is to be considered as the input to the problem we are trying to solve.

We use  $Q_{\epsilon_0, \epsilon_1}^{*,N}(\text{Ham}(N))$  to refer to the quantum time complexity of Hamiltonian cycle verification problem on network  $N$  where for any 0-input  $M$  (i.e.  $M$  is not a Hamiltonian cycle), the algorithm has to output zero with probability at least  $1 - \epsilon_0$  and for any 1-input  $M$  (i.e.  $M$  is a Hamiltonian cycle), the algorithm has to output one with probability at least  $1 - \epsilon_1$ . (We call this type of algorithm  $(\epsilon_0, \epsilon_1)$ -error.) When  $\epsilon_0 = \epsilon_1 = \epsilon$ , we simply write  $Q_\epsilon^{*,N}(\text{Ham}(N))$ . Define  $Q_{\epsilon_0, \epsilon_1}^{*,N}(\text{ST}(N))$  and  $Q_{\epsilon_0, \epsilon_1}^{*,N}(\text{Conn}(N))$  similarly.

We also study the *gap versions* of verification problems. For any integer  $\delta \geq 0$ , property  $\mathcal{P}$  and a subnetwork  $M$  of  $N$ , we say that  $M$  is  $\delta$ -far from  $\mathcal{P}$  if we have to add at least  $\delta$  edges from  $N$  and remove any number of edges in order to make  $M$  satisfy  $\mathcal{P}$ . For example,  $M$  is  $\delta$ -far from being a Hamiltonian cycle if we have to add at least  $\delta$  edges from  $N$  to  $M$  and remove any number of edges in order to make  $M$  a Hamiltonian cycle. We denote the problem of distinguishing between the case where the subnetwork  $M$  satisfies  $\mathcal{P}$  and  $\delta$ -far from satisfying  $\mathcal{P}$  the  $\delta$ - $\mathcal{P}$  problem (it is promised that the input is in these two cases). When we do not want to specify  $\delta$ , we write **Gap- $\mathcal{P}$** .

**Graph problems in the communication complexity model** To avoid confusion, we also

define graph problems on the communication complexity model here. Let  $G$  be a graph of  $n$  nodes<sup>1</sup>. We partition edges of  $G$  to  $E_A(G)$  and  $E_B(G)$ . Alice and Bob are given a set of edges  $E_A(G)$  and  $E_B(G)$ , respectively. They want to determine whether  $G$  has some property, e.g.,  $G$  is a Hamiltonian cycle ( $\text{Ham}_n$ )<sup>2</sup>, a spanning tree ( $\text{ST}_n$ ), or is connected ( $\text{Conn}_n$ ). For the purpose of this paper in proving lower bounds of distributed algorithms, we restrict that  $E_A(G)$  and  $E_B(G)$  are *perfect matchings* in the case of the Hamiltonian cycle problem.

We let  $Q_{\epsilon_0, \epsilon_1}^{*,cc}(\mathcal{P}_n)$  denote the communication complexity in the quantum setting with entanglement of graph property  $\mathcal{P}$  on  $n$ -node input graph where for any  $i$ -input (an input whose correct output is  $i$ ) the algorithm must output  $i$  with probability at least  $1 - \epsilon_i$ . We simply write  $Q_\epsilon^{*,cc}(\mathcal{P}_n)$  instead of  $Q_{\epsilon, \epsilon}^{*,cc}(\mathcal{P}_n)$ . For a closely related model called the *server model* (defined in Section 6), we use  $Q_\epsilon^{*,server}(\mathcal{P}_n)$  to denote the communication complexity.

We also consider the gap version in the case of communication complexity. The notion of  $\delta$ -far is slightly different in that we can add any edges to  $G$  instead of adding only edges in  $N$  to  $M$  as in the case of distributed network<sup>3</sup>. For example, we say that  $G$  is  $\delta$ -far from being a Hamiltonian cycle if we have to add at least  $\delta$  edges and remove any number of edges in order to make  $G$  a Hamiltonian cycle. We define the notion of  $\delta$ -far spanning tree and connected component similarly.

We mention the reason behind our complexity notations. First, we use  $*$  as in  $Q^*$  in order to emphasize that our lower bounds hold even when there is a prior entanglement, as usually done in the literature. Since we deal with different models in this paper, we put the model name after  $*$ . Thus, we have  $Q^{*,cc}$  for the case of two-party communication complexity model,  $Q^{*,server}$  for the case of server model, and  $Q^{*,N}$  for the case of distributed algorithm on a distributed network  $N$ .

## 5 Our Main Technical Results

Our main results are non-trivial lower bounds of *quantum* algorithms for fundamental graph problems on distributed networks. Our bounds hold even when nodes in the networks share an unlimited number of entangled qubits and approximated answers are allowed. Our specific results for verification and optimization graph problems are as follows.

**1. Verification problems** We prove a *tight* two-sided error quantum lower bound of  $\tilde{\Omega}(\sqrt{n})$  time, where  $n$  is the number of nodes in the distributed network and  $\tilde{\Theta}(x)$  hides  $\text{poly log } x$ , for the *Hamiltonian cycle* and *spanning tree verification problems*. Our lower bound holds even in a network of small ( $O(\log n)$ ) diameter.

**Theorem 5.1** (Verification Lower Bound). *For any  $B$  and large  $n$ , there exists  $\epsilon > 0$  and a  $B$ -model  $n$ -node network  $N$  of diameter  $\Theta(\log n)$  such that any  $(\epsilon, \epsilon)$ -error quantum algorithm with prior entanglement for Hamiltonian cycle, and spanning tree verification problems on  $N$  requires  $\Omega(\sqrt{\frac{n}{B \log n}})$  time. That is,  $Q_\epsilon^{*,N}(\text{Ham}(N))$  and  $Q_\epsilon^{*,N}(\text{ST}(N))$  are  $\Omega(\sqrt{\frac{n}{B \log n}})$ .*

Our bound implies a new bound on the classical setting which answers the open problem in [14], and is the *first randomized lower bound* for both graph problems, subsuming the deterministic lower bounds of Hamiltonian cycle verification [14], spanning tree verification [14] and minimum

<sup>1</sup>To avoid confusion, throughout the paper we use  $G$  to denote the input graph in the communication complexity model and  $N$  and  $M$  to denote the distributed network and its subnetwork, respectively, unless specified otherwise. For any graph  $H$ , we use  $V(H)$  and  $E(H)$  to denote the set of nodes and edges in  $H$ , respectively.

<sup>2</sup> $\text{Ham}_n$  is used for the Hamiltonian cycle verification problem in the communication complexity model, where  $n$  denotes the size of input graphs, and  $\text{Ham}(N)$  is used for the Hamiltonian cycle verification problem on the distributed network model, where  $N$  denotes the network we are considering.

<sup>3</sup>We note that the notion of  $\delta$ -far should not be confused with the notion of  $\epsilon$ -far usually used in property testing literature where we need to add and remove at least  $\epsilon$  fraction of edges in order to achieve a desired property. The two notions are closely related. The notion that we chose makes it more convenient to reduce between problems on different models.

	Problems	Previous results	Our results
$B$ -model distributed network	Ham, ST, MST verification	$\Omega(\sqrt{n/B \log n})$ deterministic, classical communication [14, 35]	$\Omega(\sqrt{n/B \log n})$ two-sided error, quantum communication with entanglement
	Conn and other verification problems	$\Omega(\sqrt{n/B \log n})$ two-sided error, classical communication [14]	
	$\alpha$ -approx MST and other problems	$\Omega(\sqrt{n/B \log n})$ Monte Carlo, classical communication for $W = \Omega(\alpha n)$ [14]	$\Omega(\min(\sqrt{n}, W/\alpha)/\sqrt{B \log n})$ Monte Carlo, quantum communication with entanglement
communication complexity	Ham, ST, and other verification problems	$\Omega(n)$ one-sided error, classical communication [57]	$\Omega(n)$ two-sided error, quantum communication with entanglement
	Gap-Ham, Gap-ST, Gap-Conn, and other gap problems for $\Omega(n)$ gap	unknown	$\Omega(n)$ one-sided error, quantum communication with entanglement

Figure 1: Previous and new lower bounds. We note that  $n$  is the number of nodes in the network in the case of distributed network and the number of nodes in the input graph in the case of communication complexity.

spanning tree verification [35]. It is also shown in [14] that **Ham** can be reduced to several problems via deterministic classical-communication reductions. Since these reductions can be simulated by quantum protocols, we can use these reductions straightforwardly to show that all lower bounds in [14] hold even in the quantum setting.

**Corollary 5.2.** *For any  $B$  and large  $n$ , there exists  $\epsilon > 0$  and a  $B$ -model  $n$ -node network  $N$  of diameter  $\Theta(\log n)$  such that any  $(\epsilon, \epsilon)$ -error quantum algorithm with prior entanglement for the following verification problems on  $N$  requires  $\Omega(\sqrt{\frac{n}{B \log n}})$  time: Connected component, minimum spanning tree, spanning connected subgraph, cycle containment,  $e$ -cycle containment, bipartiteness,  $s$ - $t$  connectivity, connectivity, cut, edge on all paths,  $s$ - $t$  cut and least-element list.*

We refer to Section 12.2 for definitions. Fig. 1 compares our results with previous results for various verification problems.

**2. Optimization problems** We show a *tight*  $\Omega(\min(W/\alpha, \sqrt{n}))$ -time lower bound for any  $\alpha$ -approximation quantum randomized (Monte Carlo) distributed algorithm for the MST problem.

**Theorem 5.3** (Optimization Lower Bound). *For any  $n, B, W$  and  $\alpha < W$  there exists  $\epsilon > 0$  and a  $B$ -model  $\Theta(n)$ -node network  $N$  of diameter  $\Theta(\log n)$  such that any  $\epsilon$ -error  $\alpha$ -approximation quantum algorithm with prior entanglement for computing the minimum spanning tree problem on  $N$  with weight function  $w : E(N) \rightarrow \mathbb{R}_+$  such that  $\frac{\max_{e \in E(N)} w(e)}{\min_{e \in E(N)} w(e)} \leq W$  requires  $\Omega(\frac{1}{\sqrt{B \log n}} \min(W/\alpha, \sqrt{n}))$  time.*

Our results generalize the bounds in [14] to the quantum setting. Moreover, this lower bound, when restricted to the classical communication setting, improves the bound of [14] for the case of small  $W$  and matches the deterministic upper bound of  $\Omega(\min(W/\alpha, \sqrt{N}))$  resulting from a combination of Elkin’s  $\alpha$ -approximation  $O(W/\alpha)$ -time deterministic algorithm [22] and Peleg and Rubinfeld’s  $O(\sqrt{N})$ -time exact deterministic algorithm [25, 41] in the classical communication model. Thus it is the first bound that is *tight for all values of  $W$* . Fig. 2 compares our lower bounds with previous lower bounds.

Again, using the same reduction as in [14], our bound also implies that all lower bounds in [14] hold even in the quantum setting.

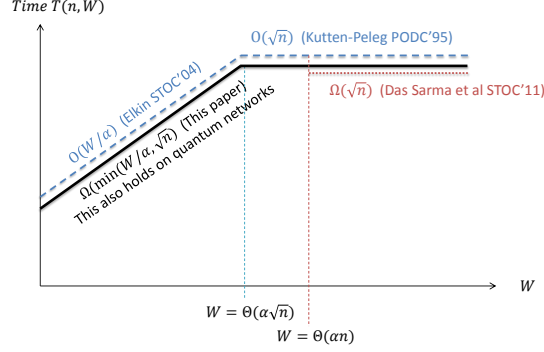


Figure 2: Previous and our bounds for approximating the MST problem in distributed networks when  $N$  and  $\alpha$  are fixed. The dashed line (in red) represents the deterministic upper bounds (Algorithms). The dotted line (in red) is the previous lower bound of randomized (Monte Carlo) algorithms. The solid line (in black) represents the bounds shown in this paper. Note that the previous lower bounds hold only in the classical setting while the new lower bounds hold in the quantum setting even when entanglement is allowed. See Theorem 5.3 and Corollary 5.4 for the full statements.

**Corollary 5.4** (Main results for optimization problems). *For any  $n, B, W$  and  $\alpha$  there exists  $\epsilon > 0$  and a  $B$ -model  $\Theta(n)$ -node network  $N$  of diameter  $\Theta(\log n)$  such that any  $\epsilon$ -error  $\alpha$ -approximation quantum algorithm with prior entanglement for computing the following problems on  $N$  with weight function  $w : E(N) \rightarrow \mathbb{R}_+$  such that  $\frac{\max_{e \in E(N)} w(e)}{\min_{e \in E(N)} w(e)} \leq W$  requires  $\Omega(\frac{1}{\sqrt{B \log n}} \min(W/\alpha, \sqrt{n}))$  time: minimum spanning tree, shallow-light tree,  $s$ -source distance, shortest path tree, minimum routing cost spanning tree, minimum cut, minimum  $s$ - $t$  cut, shortest  $s$ - $t$  path and generalized Steiner forest.*

We refer to Section 12.2 for definitions of the above optimization problems.

### 5.1 Additional results: Communication complexity lower bounds

In proving the main results, we prove some new lower bounds in the two-party quantum communication complexity model. In fact our bounds hold for a slightly stronger model called the *server model*, which will be introduced in Section 6.

First, we show a tight bound of the Hamiltonian cycle in the two-sided error quantum setting on  $n$ -node input graphs.

**Theorem 5.5.** *For any  $n$  and some  $\epsilon > 0$ ,  $Q_{\epsilon}^{*,cc}(\text{Ham}_n) = \Omega(n)$ .*

To the best of our knowledge, this the first two-sided error lower bounds of  $\text{Ham}_n$  even in the classical setting (nondeterministic (thus one-sided error) lower bounds are previously known [57]). As a corollary, we immediately get lower bounds for many other verification problems. Of these the lower bound for Spanning tree is new, whereas the bounds for Bipartiteness and  $s$ - $t$  connectivity follow from a reduction from Inner Product given in [2], and a lower bound for Connectivity was recently shown in [28] (but does not show hardness of Hamiltonian cycle).

**Corollary 5.6.** *For any  $n$  and some  $\epsilon > 0$ ,  $Q_{\epsilon}^{*,cc}(\text{P}_n) = \Omega(n)$ , where  $\text{P}_n$  can be any of the following verification problems: spanning tree, connectivity,  $s$ - $t$  connectivity, and bipartiteness.*

We also extend our study to the *gap version* of the graph verification problems in the one-sided error setting via a reduction from recent lower bounds in [32].

**Theorem 5.7.** *For any  $n$ , some constants  $\epsilon, \beta > 0$ ,  $Q_{0,\epsilon}^{*,cc}((\beta n) - \text{Ham}_n) = \Omega(n)$ .*



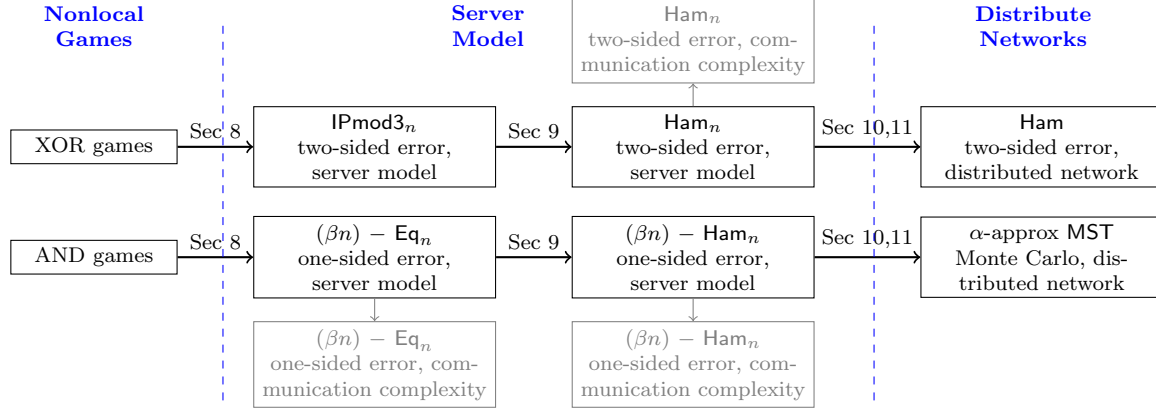


Figure 3: Our proof structure. Lines in gray show the implications of our results in communication complexity.

We note that in proving the above theorem we observe lower bounds of the gap version of Set Disjointness and Equality. As before, the theorem implies lower bounds of other problems.

**Corollary 5.8.** *For any  $n$ , some constants  $\epsilon, \beta > 0$ ,  $Q_{0,\epsilon}^{*,cc}((\beta n) - P_n) = \Omega(n)$ , where  $P_n$  can be any of the following verification problems: spanning tree, connectivity,  $s$ - $t$  connectivity and bipartiteness.*

## 6 Overview of Technical Approach

The high-level idea behind our lower bound proofs is by establishing a connection between quantum communication complexity and quantum distributed network computing. Our work is inspired by [14] (following a line of work in [55, 47, 22, 35]) which shows lower bounds of many graph verification and optimization problems in the *classical* distributed computing model. The main technique used to show the classical lower bounds in [14] is the *Simulation Theorem* (cf. Theorem 3.1 in [14]) which shows how one can use lower bounds for problems in the standard two-party (Yao’s) classical communication complexity model [40] to derive lower bounds of equivalent problems in the “distributed” version of communication complexity. Specifically, the Simulation Theorem relates the *communication* lower bound from the standard two-party communication complexity model to compute some appropriately chosen function  $f$  (e.g., Set Disjointness or Equality) to the distributed *time* complexity lower bound for computing the same function in a specially chosen graph  $G$ . They use a graph that was first used in [55]. The Simulation Theorem effectively reduces the task of proving lower bounds of *graph verification* problems to merely picking the right function  $f$  to reduce from. Two functions that were used in [14] were Set Disjointness and Equality, which are quintessential problems in the world of communication complexity [40]. Finally, one can reduce the verification problem to hardness of distributed approximation for a variety of problems to show that the same lower bounds hold for approximation algorithms as well. Our starting point for showing lower bounds for the quantum setting is a question whether there is a *quantum* version of the Simulation Theorem, i.e., whether one can prove quantum distributed algorithm lower bounds via quantum (two-party) communication complexity lower bounds. However, this seems challenging to do due to several difficulties (as explained below); in fact, it leads us to a proof that is substantially different compared to the classical setting. In particular, while we cannot prove a “quantum” Simulation Theorem using the standard two-party model, we manage to identify a new communication model called the *server* communication model and prove a server model version of the Simulation Theorem. Our new Simulation Theorem shows that the server model captures the hardness of distributed algorithms well in both the classical and quantum setting. We then show that many important bounds in the standard quantum two-party model also hold in the server model. Some

of the bounds we prove are even new in the standard model and classical setting. These bounds together with our new Simulation Theorem lead to our main results. Details are as follows.

**Where the Simulation Theorem Fails and the Server Model Saves Us** The first difficulty in extending previous techniques is that it is not clear whether the Simulation Theorem holds in the quantum setting (i.e., whether a quantum two-party communication complexity implies a distributed algorithm lower bound). The main reason is that its proof seems to require the two players, say Alice and Bob, to simulate some common nodes in a network. While this is easy to do in the classical setting, it is not clear whether it is still possible in the quantum setting.

To illustrate this point, let us introduce a new model called the *server* model. There are three players in the server model, Carol, David and the server. Carol and David receive the inputs  $x$  and  $y$ , respectively, and want to compute  $f(x, y)$  for some function  $f$ . As in the two-party model, Carol and David can talk to each other. Additionally, they can talk to the server. The catch here is that the server can send messages for *free*. Thus, the communication complexity in the server model counts only messages sent by Carol and David. This model, even without prior entanglement, is clearly at least as strong as the standard quantum communication complexity model with shared entanglement, since the server can dispense any entangled state to Carol and David.

With the server model in our perspective, we can refine the previous connection between two-party communication complexity and distributed algorithm lower bound (i.e., the Simulation Theorem) to two separate connections: (1) A connection between the two-party and server models and (2) A connection between the server model and distributed networks. We will show that the second connection holds even in the quantum setting with entanglement:

**Theorem 6.1** (Details in Section 10). *For any  $B, L, \Gamma \geq \log L, \beta \geq 0$  and  $\epsilon_0, \epsilon_1 > 0$ , there exists a  $B$ -model quantum network  $N$  of diameter  $\Theta(\log L)$  and  $\Theta(\Gamma L)$  nodes such that if  $Q_{\epsilon_0, \epsilon_1}^{*, N}((\beta\Gamma)\text{-Ham}(N)) \leq \frac{L}{2} - 2$  then  $Q_{\epsilon_0, \epsilon_1}^{*, \text{server}}((\beta\Gamma)\text{-Ham}_\Gamma) = O((B \log L) Q_{\epsilon_0, \epsilon_1}^{*, N}((\beta\Gamma)\text{-Ham}(N)))$ .*

This theorem is similar to the Simulation Theorem in [14] except that it replaces the standard two-party model by the server model. Our proof follows from previous techniques [55, 47, 22, 35, 14, 49] (e.g., we use a similar graph and maintain the state of similar sets of nodes) but also provides some simplification (e.g., we reduce from graph problems directly, leaving most of the “hard part” in the communication complexity model, and we do not require simulating common nodes which is a source of confusion). This proof also shows that the underlying idea of this technique is in fact the same as the simple idea of *bisection width* commonly used in distributed networking as well as other areas (see, e.g., [40, 44, 39, 23]).

Now we know that the second connection holds even in the quantum setting. Thus, to prove the quantum version of the Simulation Theorem, it remains to show that the server-model communication complexity is lower bounded by the two-party communication complexity. We note that it is clear that the converse is true: the communication complexity of the server model is *at most* the two-party model in both classical and quantum settings since Carol and David can pretend to be Alice and Bob. On the other hand, it is conceivable the two models are equivalent: Since the server does not get any input, he seems to act only as a channel. Are these two models equivalent? A partial answer to this question leads to one of the difficulties in the quantum setting: *While it is easy to prove that the two models are equivalent in the classical setting, it is not clear if they are in the quantum setting.*

To explain some issues in the quantum setting, let us sketch the proof of this in the classical setting. Let us first consider the deterministic setting. The proof is by the following “simulation” argument. Alice will simulate Carol and the server. Bob will simulate David and the server. In each round, Alice will see all messages sent from the server to Carol and thus she can keep simulating Carol. However, she does not see the message sent from David to the server which she needs to

simulate the server. So, she must get this message from Bob. Similarly, Bob will get from Alice the message sent from Carol to the server. These are the only messages we need in each round in order to be able to simulate the protocol. Observe that the total number of bits sent between Alice and Bob is exactly the number of bits sent by Carol and David to the server. Thus, the complexities of both models are exactly the same in the deterministic case. We can conclude the same thing for the public coin setting (where all parties share a random string) since Alice and Bob can use their shared coin to simulate the shared coin of Carol, David and the server.

The above argument, however, does not seem to work in the quantum setting. The main issue with a simulation along the lines of the one sketched above is that Alice and Bob cannot simulate a “copy” of the server each. For instance one could try to simulate the server’s state in a distributed way by maintaining the state that results by applying CNOT to every qubit of the server and a fresh qubit, and distribute these qubits to Alice and Bob. But then if the server sends a message to Carol, Bob would have to disentangle the corresponding qubits in his copy, which would require a message to Alice.

While we leave as an open question whether the two models are equivalent in the quantum setting, we prove that many lower bounds in the two-party model extend to the server model.

**Server-model Communication Complexity** (Details in Section 8 and 9) In this paper we show lower bounds of two types of problems. The first type consists of basic problems studied in the standard two-party model such as Equality and Set Disjointness. To show lower bounds of these problems, we show a connection between the server model and *nonlocal games*. Nonlocal games are games where Alice and Bob receive an input  $x$  and  $y$  from some distribution and want to compute  $f(x, y)$ . However, they cannot talk to each other. Instead, they output one bit, say  $a$  and  $b$ , which are then combined to be an output. For example, in *XOR games* and *AND games*, these bits are combined as  $a \oplus b$  and  $a \wedge b$ , respectively. The players’ goal is to maximize the probability that the output is  $f(x, y)$ . We relate nonlocal games to the server model by showing that the XOR- and AND-game players can use an efficient server-model protocol to guarantee a good winning chance.

**Lemma 6.2** (Details in Section 8). *For any boolean function  $f$  and  $\epsilon_0, \epsilon_1 \geq 0$ , there is an XOR-game strategy  $\mathcal{A}'$  and AND-game strategy  $\mathcal{A}''$  such that, for any input  $(x, y)$ , with probability  $4^{-2Q_{\epsilon_0, \epsilon_1}^{*, \text{server}}(f)}$ ,  $\mathcal{A}'$  and  $\mathcal{A}''$  output  $f(x, y)$  with probability at least  $1 - \epsilon_{f(x, y)}$ ; otherwise  $\mathcal{A}'$  outputs 0 and 1 with probability  $1/2$  each, and  $\mathcal{A}''$  outputs 0 with probability 1.*

Following the  $\gamma_2$ -norm techniques in [46, 60, 43] and the recent method of [32], we show one- and two-sided error lower bounds of many problems on the server model, which match the two-party model lower bounds.

The second type of problems are graph problems. To show a lower bound of Hamiltonian cycle verification ( $\text{Ham}_n$ ), we derive reductions (using some novel gadgets) from *Inner Product mod 3* ( $\text{IPmod3}_n$ ) and *Gap Equality* ( $\text{Gap-Eq}_n$ ), defined in Section 8. This result also implies a lower bound for spanning tree ( $\text{ST}_n$ ) verification. (We note again that the lower bounds of  $\text{Ham}_n$  and  $\text{ST}_n$  are new even in the classical two-party model.) Using the server-model communication complexity of  $\text{Ham}_n$  and  $\text{Gap-Ham}_n$ , we can show lower bounds of distributed algorithms as discussed earlier. Fig. 3 depicts our proof strategy.

## 7 Conclusion

In this paper, we derive the first non-trivial lower bounds for important network problems in a quantum distributed network. Our approach gives a uniform and powerful way to prove lower bounds for various problems. Using our approach it might be possible to prove lower bounds for other problems or possibly strengthen existing lower bounds for problems such as shortest path and minimum cut (which are not tight at present). Other interesting open problems are: (1) Can we

derive a “Quantum” two-party version of the Simulation Theorem? In other words, can we relate distributed algorithm lower bounds to the two-party quantum communication complexity model instead of the server model? This will be very helpful as it can simplify the proofs by using existing bounds in the two-party model. (2) Finally, it will be interesting to explore *upper* bounds in the quantum setting as well: Do quantum distributed algorithms help in solving fundamental graph problems such as shortest path and minimum cut?

# Appendix

## Part I: Proofs

In this part, we explain our proofs at a high level. Some details are postponed to Part II for readability. This part is organized as follows (also see Fig. 3). In Section 8 we show server-model lower bounds of two basic problems in communication complexity called Inner Product mod 3 and Gap-Equality. In Section 9, we show server-model lower bounds of the Hamiltonian cycle problem and its gap version by reducing from the above mentioned two basic problems. In Section 10, we prove a quantum version of the Simulation Theorem which shows that a server-model lower bound implies a quantum distributed network lower bound. Finally in Section 11 we combine the results to show our main theorems.

### 8 Server-Model complexity of $\text{IPmod3}_n$ and $\text{Gap-Eq}_n$

In this section we prove lower bounds of the following problems. In the *Inner Product mod 3* problem (denoted by  $\text{IPmod3}_n$ ), Carol and David are given  $n$ -bit strings  $x$  and  $y$  respectively. They have to output 1 if  $(\sum_{i=1}^n x_i y_i) \bmod 3 = 0$  and 0 otherwise. In the *Gap Equality* problem with parameter  $\delta$  (denoted by  $\delta\text{-Eq}_n$ ), Carol and David also receive  $n$ -bit strings  $x$  and  $y$ . Additionally, it is *promised* that either  $x = y$  or the hamming distance between  $x$  and  $y$ , denoted by  $\Delta(x, y)$  is more than  $\delta$ , i.e.  $|\{i \mid x_i \neq y_i\}| > \delta$ . In the former case, they have to output 1; otherwise they have to output 0. Details of this section such as formal definitions and proofs can be found in Section 13.

**Theorem 8.1.** *For some  $\beta, \epsilon > 0$  and any large  $n$ ,  $Q_{\epsilon}^{*, \text{server}}(\text{IPmod3}_n)$  and  $Q_{0, \epsilon}^{*, \text{server}}((\beta n)\text{-Eq}_n)$  are  $\Omega(n)$ .*

Our proof makes use of the relationship between the server model and *nonlocal games*. In such games, Alice and Bob receive input  $x$  and  $y$  from some distribution  $\pi$  that is known to the players. As usual they want to compute a boolean function  $f(x, y)$  such as Equality or Inner Product mod 3. However, they cannot communicate to each other. Instead, each of them can send one bit, say  $a$  and  $b$ , to a referee. The referee then combines  $a$  and  $b$  using some function  $g$  to get an output of the game  $g(a, b)$ . The goal of the players is to come up with a strategy (which could depend on distribution  $\pi$  and function  $g$ ) that maximizes the probability that  $g(a, b) = f(x, y)$ . We call this the *winning probability*. One can define different nonlocal games based on what function  $g$  the referee will use. Two games of our interest are *XOR*- and *AND*-games where  $g$  is *XOR* and *AND* functions, respectively.

Our proof follows the framework of proving two-party quantum communication complexity lower bounds via nonlocal games (see, e.g., [42, 43, 32]). The main modification is the following lemma which shows that the XOR- and AND-game players can make use of an efficient server-model protocol to guarantee a good winning probability.

**Lemma 8.2.** *For any boolean function  $f$  and  $\epsilon_0, \epsilon_1 \geq 0$ , there is an XOR-game strategy  $\mathcal{A}'$  and AND-game strategy  $\mathcal{A}''$  such that, for any input  $(x, y)$ , with probability  $4^{-2Q_{\epsilon_0, \epsilon_1}^{*, \text{server}}(f)}$ ,  $\mathcal{A}'$  and  $\mathcal{A}''$  output  $f(x, y)$  with probability at least  $1 - \epsilon_{f(x, y)}$ ; otherwise  $\mathcal{A}'$  outputs 0 and 1 with probability  $1/2$  each, and  $\mathcal{A}''$  outputs 0 with probability 1.*

*Proof Sketch.* We prove the lemma in a similar way to the proof of Theorem 5.3 in [42] (attributed to Buhrman). Let  $\mathcal{A}$  be any  $(\epsilon_0, \epsilon_1)$ -error server-model protocol with communication complexity  $T$ . We will construct  $\mathcal{A}'$  and  $\mathcal{A}''$  that simulate  $\mathcal{A}$ . First we simulate  $\mathcal{A}$  with an additional assumption that there is a “fake server” that sends messages in the nonlocal games, but the two players in the games (Alice and Bob) do not send any message to the fake server. Later we will eliminate this fake server. We will refer to parties in the server model as Carol, David and the *real* server.

Using teleportation (see, e.g., [51]), it can be assumed that Carol and David send  $2T$  classical bits to the real server instead of sending  $T$  qubits (the server can set up the necessary entanglement for free). Assume that, on an input  $(x, y)$ , Carol and David send bits  $c_t$  and  $d_t$  in the  $t^{\text{th}}$  round, respectively. (We note one detail here that in reality  $c_t$  and  $d_t$ , for all  $t$ , are random variables. We will ignore this fact here to illustrate the main idea.)

Now, Alice, Bob and the fake server generate shared random strings  $a_1 \dots a_t$  and  $b_1 \dots b_t$  (this can be done since their states are entangled). These strings serve as a “guessed” communication sequence of  $\mathcal{A}$ . Alice, Bob and the fake protocol simulate Carol, David and the real protocol, respectively. However, in each round  $t$ , instead of sending bit  $c_t$  that Carol sends to the real server, Alice simply looks at  $a_t$  and continues playing if her guessed communication is the same as the real communication, i.e.  $c_t = a_t$ . Otherwise, she “aborts”: In the XOR-game protocol  $\mathcal{A}'$  she outputs 0 and 1 with probability  $1/2$  each, and in the AND-game protocol  $\mathcal{A}''$  she outputs 0. Bob does the same thing with  $d_t$  and  $b_t$ .

The fake server simply assumes it receives  $a_t$  and  $b_t$  and continues sending messages to Alice and Bob. Observe that the probability of never aborting is  $4^{-T}$  (i.e., when the random strings  $a_1 \dots a_T$  and  $b_1 \dots b_T$  are the same as the communication sequences  $c_1 \dots c_T$  and  $d_1 \dots d_T$ , respectively). If no one aborts, Alice will output Carol’s output while Bob will output 0 in XOR-game protocol  $\mathcal{A}'$  and 1 in AND-game protocol  $\mathcal{A}''$ . If no one aborts, Alice, Bob and the fake server perfectly simulate  $\mathcal{A}$  and thus output  $f(x, y)$  with probability at least  $1 - \epsilon_{f(x, y)}$  in both protocols. Otherwise (with probability at most  $1 - 4^{-T}$ ) one or both players will abort and the output will be randomly 0 and 1 in  $\mathcal{A}'$  and 0 in  $\mathcal{A}''$ . This is exactly what we claim in the theorem except that there is a fake server.

Now we eliminate the fake server. Notice that the fake server never receives anything from Alice and Bob. Hence we can assume that the fake server sends all his messages to Alice and Bob before the game starts (before the input is given), and those messages can be viewed as prior entanglement. We thus get standard XOR- and AND-game strategies without a fake server.  $\square$

Now we sketch the proof of Theorem 8.1. To show that  $Q_\epsilon^{*, \text{server}}(\text{IPmod}3_n) = \Omega(n)$ , we use an XOR-game strategy  $\mathcal{A}'$  and  $\epsilon_0 = \epsilon_1 = \epsilon$  from Lemma 8.2. Using this we can extend the theorem of Linial and Shraibman [46] from the two-party model to the server model and show that  $Q_\epsilon^{*, \text{server}}(f)$  is lower bounded by an *approximate*  $\gamma_2$  norm:  $Q_\epsilon^{*, \text{server}}(f) = \Omega(\log \gamma_2^{2\epsilon}(A_f))$  for some matrix  $A_f$  defined by  $f$ . (See Lemma 13.2 in Section 13 for detail.)

Using  $f = \text{IPmod}3_n$ , one can then extend the proof of Lee and Zhang [43, Theorem 8] to lower bound approximate  $\gamma_2$  norm by an *approximate degree* of some certain functions  $f'$  (see Lemma 13.4 in Section 13 for detail), and follow the proof of Sherstov [60] and Razborov [58] to finally prove that

$$Q_\epsilon^{*, \text{server}}(\text{IPmod}3_n) = \Omega(\log \gamma_2^{2\epsilon}(A_{\text{IPmod}3_n})) = \Omega(\deg_{2\epsilon}(f')) = \Omega(n).$$

We note that this technique actually extends all lower bounds we are aware of on the two-party model to the server model (e.g. those in [58, 60, 43]).

To prove that  $Q_{0, \epsilon}^{*, \text{server}}((\beta n)\text{-Eq}_n) = \Omega(n)$  for some  $\beta, \epsilon > 0$ , we use an AND-game strategy  $\mathcal{A}''$  with  $\epsilon_0 = 0$  and  $\epsilon_1 = \epsilon = 1/2$  from Lemma 8.2. We adapt a recent result by Klauck and de Wolf [32], which shows that  $Q_{0, 1/2}^{*, \text{cc}}(f) \geq (\log \text{fool}^1(f) - 1)/2$ . Here  $\text{fool}^1(f)$  refers to the 1-fooling set size of  $f$ , which is defined as follows. A 1-fooling set for  $f$  is a set  $F = \{(x, y)\}$  of input pairs with the following properties.

- If  $(x, y) \in F$  then  $f(x, y) = 1$
- If  $(x, y), (x', y') \in F$  then  $f(x, y') = 0$  or  $f(x', y) = 0$

The lower bound in [32] actually applies to AND-games as follows. Suppose Alice and Bob receive inputs  $x, y$ , perform local measurements on a shared entangled state, and output bits  $a, b$ .

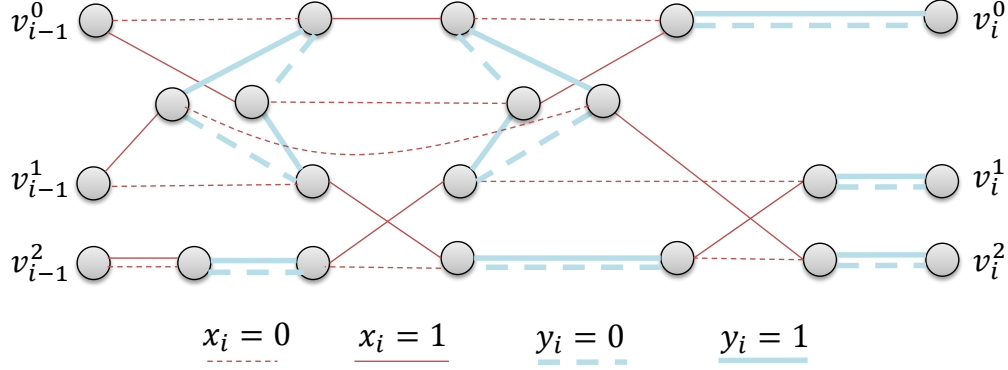


Figure 4: The construction of gadget  $G_i$ . If  $x_i = 0$  then Alice adds dashed thin edges (in red); otherwise she adds solid thin edges (in red). If  $y_i = 0$  then Bob adds dashed thick edges (in blue); otherwise he adds solid thick edges (in blue). (See Fig. 5 for pictures of  $G_i$  for all possible values of  $x_i$  and  $y_i$ .)

Then the probability that  $a \wedge b = 1$  for a uniformly random  $x, y \in F$  is at most  $1/\text{fool}^1(f)$ , if the probability that  $a \wedge b = 1$  for  $x, y$  with  $f(x, y) = 0$  is always 0.

Lemma 8.2 for the case of AND-games implies that there is an AND-game strategy  $\mathcal{A}''$  such that if  $f(x, y) = 0$  then  $\mathcal{A}''$  always output 0 and if  $f(x, y) = 1$  then  $\mathcal{A}''$  outputs 1 with probability at least  $(1 - \epsilon)4^{-2Q_{0,\epsilon}^{*,\text{server}}(f)}$ . This implies that  $(1 - \epsilon)4^{-2Q_{0,\epsilon}^{*,\text{server}}(f)} \leq 1/\text{fool}^1(f)$ . In other words, if  $\text{fool}^1(f) = 2^{\Omega(n)}$  then  $Q_{0,1/2}^{*,\text{server}}(f) = \Omega(n)$ .

All that remains is to define a good fooling set for  $(\beta n)$ -Eq $_n$ . Fix any  $1/4 > \beta > 0$ . The idea is to use a good error-correcting code to construct the fooling set. Let  $\Delta(x, y)$  denote the Hamming distance. Let  $C$  be a set of  $n$ -bit strings such that the Hamming distance between any distinct  $x, y \in C$  is at least  $2\beta n$ . Due to the Gilbert-Varshamov bound such codes  $C$  exist with  $|C| \geq 2^{(1-H(2\beta))n} = 2^{\Omega(n)}$ , where  $H$  denotes the binary entropy function. Hence we can conclude that  $Q_{0,1/2}^{*,\text{server}}((\beta n)\text{-Eq}_n) = \Omega(n)$ .

## 9 Server-model communication complexity of Ham $_n$

This section is devoted to proving the following theorem.

**Theorem 9.1.** *For any  $n$  and some constants  $\epsilon, \beta > 0$ ,  $Q_\epsilon^{*,\text{server}}(\text{Ham}_n)$  and  $Q_{0,\epsilon}^{*,\text{server}}((\beta n)\text{-Ham}_n)$  are  $\Omega(n)$ .*

We first sketch the lower bound proof of  $Q_\epsilon^{*,\text{server}}(\text{Ham}_n)$  and show later how to extend to the gap version. More detail can be found in Section 14. We will show that for any  $0 \leq \epsilon \leq 1$  and some constant  $c$ ,  $Q_\epsilon^{*,\text{server}}(\text{IPmod}3_n) = O(Q_\epsilon^{*,\text{server}}(\text{Ham}_{cn}))$ . The theorem then immediately follows from the fact that  $Q_\epsilon^{*,\text{server}}(\text{IPmod}3_n) = \Omega(n)$  (cf. Theorem 8.1).

Let  $x = x_1 \dots x_n$  and  $y = y_1 \dots y_n$  be the input of  $\text{IPmod}3_n$ . We construct a graph  $G$  which is an input of  $\text{Ham}_{cn}$  as follows. The graph  $G$  consists of  $n$  gadgets, denoted by  $G_1, \dots, G_n$ . For any  $1 \leq i \leq n - 1$ , gadgets  $G_i$  and  $G_{i+1}$  share exactly three nodes denoted by  $v_i^0, v_i^1, v_i^2$ . Each gadget  $G_i$  is constructed based on the values of  $x_i$  and  $y_i$  as outlined in Fig. 4. The following observation can be checked by drawing  $G_i$  for all cases of  $x_i$  and  $y_i$  (as in Fig. 5).

**Observation 9.2.** *For any value of  $(x_i, y_i)$ ,  $G_i$  consists of three paths where  $v_{i-1}^j$  is connected by a path to  $v_i^{(j+x_i \cdot y_i) \bmod 3}$ , for any  $0 \leq j \leq 2$ . Moreover, Alice's (respectively Bob's) edges, i.e. thin (red) lines (respectively thick (blue) lines) in Fig. 4, form a matching that covers all nodes except  $v_i^j$  (respectively  $v_{i-1}^j$ ) for all  $0 \leq j \leq 2$ .*

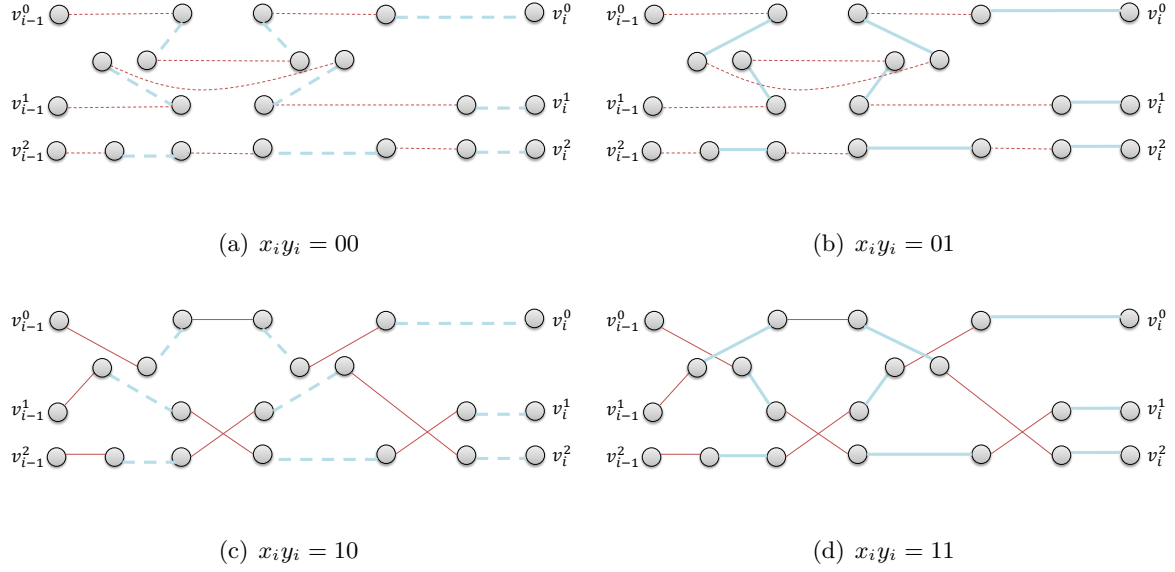


Figure 5: Gadget  $G_i$  for different values of  $x_i$  and  $y_i$ . The main observation is that if  $x_i \cdot y_i = 0$  then  $G_i$  consists of paths from  $v_{i-1}^j$  to  $v_i^j$  for all  $0 \leq j \leq 2$ . Otherwise, it consists of paths from  $v_{i-1}^j$  to  $v_i^{(j+1) \bmod 3}$ .

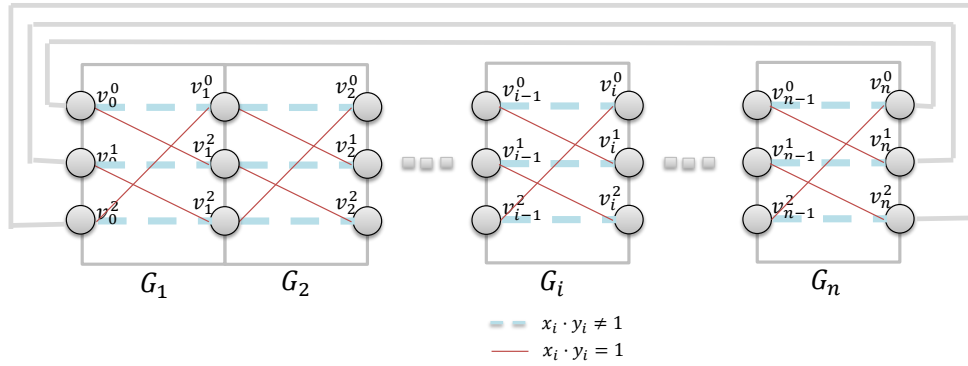


Figure 6: The graph  $G$  consists of gadgets  $G_1, \dots, G_n$ . The solid thick edges (in gray) linking between  $v_0^j$  and  $v_n^j$ , for  $0 \leq j \leq 2$  represent the fact that  $v_0^j = v_n^j$ . Lines that appear in each gadget  $G_i$  depicts what we observe in Observation 9.2: solid thin lines (in red) represent paths that will appear in  $G_i$  if  $x_i \cdot y_i = 0$ , and dashed thick lines (in blue) represent paths that will appear in  $G_i$  if  $x_i \cdot y_i = 1$ .



Thus, when we put all gadgets together, graph  $G$  will consist of three paths connecting between nodes in  $\{v_0^j\}_{0 \leq j \leq 2}$  on one side and nodes in  $\{v_n^j\}_{0 \leq j \leq 2}$  on the other. How these paths look like depend on the structure of each gadget  $G_i$  which depends on the value of  $x_i \cdot y_i$ . The following lemma follows trivially from Observation 9.2.

**Lemma 9.3.**  *$G$  consists of three paths  $P^0$ ,  $P^1$  and  $P^2$  where for any  $0 \leq j \leq 2$ ,  $P^j$  has  $v_0^j$  as one end vertex and  $v_n^{(j + \sum_{1 \leq i \leq n} x_i \cdot y_i) \bmod 3}$  as the other.*

Now, we complete the description of  $G$  by letting  $v_0^j = v_n^j$  for all  $0 \leq j \leq 2$ . It then follows that  $G$  is a Hamiltonian cycle if and only if  $\sum_{1 \leq i \leq n} x_i \cdot y_i \bmod 3 \neq 0$  (see Fig. 6; also see Lemma 14.3 and Fig. 12 in Section 14). Thus we can check that  $\sum_{1 \leq i \leq n} x_i \cdot y_i \bmod 3$  is zero or not by checking whether  $G$  is a Hamiltonian cycle or not. Theorem 9.1 now follows from Theorem 8.1.

To show a lower bound of  $Q_{0,\epsilon}^{*,server}((\beta n)\text{-Ham}_n)$ , we reduce from  $(\beta n)\text{-Eq}_n$  in a similar way using gadget  $G_i$  shown in Fig. 7. For any  $1 \leq i \leq n-1$ , gadget  $G_i$  and  $G_{i+1}$  share  $v_i^0$  and  $v_i^1$ , and we let  $v_0^0 = v_0^1$  and  $v_n^0 = v_n^1$ . It is straightforward to show that if  $x = y$  then  $G$  is a Hamiltonian cycle and if  $x_{i_j} \neq y_{i_j}$  for some  $i_1 < i_2 < \dots < i_\delta$  then  $G$  will consist of  $\delta$  cycles where each cycle starts at gadget  $G_{i_j}$  and ends at gadget  $G_{i_{j+1}}$ . We note that our reduction also gives a simplification of the rather complicated reduction in [14, Section 6].

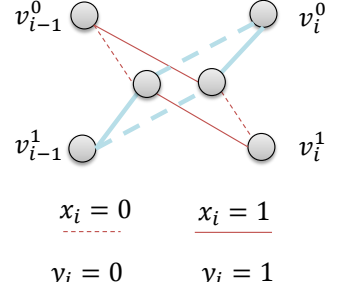


Figure 7: Gadget  $G_i$ .

## 10 From Server-Model to $B$ -Model Lower Bounds

In this section, we show that in the quantum setting, a server-model lower bound implies a  $B$ -model lower bound, as in the following theorem.

**Theorem 10.1.** *For any  $B, L, \Gamma \geq \log L$ ,  $\beta \geq 0$  and  $\epsilon_0, \epsilon_1 > 0$ , there exists a  $B$ -model quantum network  $N$  of diameter  $\Theta(\log L)$  and  $\Theta(\Gamma L)$  nodes such that if  $Q_{\epsilon_0, \epsilon_1}^{*,N}((\beta \Gamma)\text{-Ham}(N)) \leq \frac{L}{2} - 2$  then  $Q_{\epsilon_0, \epsilon_1}^{*,server}((\beta \Gamma)\text{-Ham}_\Gamma) = O((B \log L) Q_{\epsilon_0, \epsilon_1}^{*,N}((\beta \Gamma)\text{-Ham}(N)))$ .*

In words, the above theorem states that if there is an  $(\epsilon_0, \epsilon_1)$ -error quantum distributed algorithm that solves the Hamiltonian cycle verification problem on  $N$  in at most  $(L/2) - 2$  time, i.e.  $Q_{\epsilon_0, \epsilon_1}^{*,N}(\text{Ham}(N)) \leq (L/2) - 2$ , then the  $(\epsilon_0, \epsilon_1)$ -error communication complexity in the server model of the Hamiltonian cycle problem on  $\Gamma$ -node graphs is  $Q_{\epsilon_0, \epsilon_1}^{*,server}(\text{Ham}_\Gamma) = O((B \log L) Q_{\epsilon_0, \epsilon_1}^{*,N}(\text{Ham}(N)))$ . The same statement also holds for its gap version. We note that the above theorem can be extended to a large class of graph problems with some certain properties. We state it for only  $\text{Ham}$  for simplicity.

*Proof idea of Theorem 10.1.* We give the proof idea here and provide full detail in Section 15. Although we recommend the readers to read this before the full proof and believe that it is enough to reconstruct the full proof, this proof idea can be skipped without loss of continuity.

We note again that the main idea of this theorem essentially follows the ideas developed in a line of work in [55, 22, 47, 35, 14]. In particular, we construct a network following ideas in [55, 22, 47, 35, 14], and our argument is based on simulating the network by the three players of the server model. This idea follows one of many ideas implicit in the proof of the Simulation Theorem in [14] which shows how two players can simulate some class of networks. However, as we noted earlier, the previous proof does not work in the quantum setting, and it is still open whether the Simulation Theorem holds in the quantum setting. We instead use the server model. Another difference is that we prove the theorem for graph problems instead of problems on strings (such as Equality or Disjointness). This leads to some simplified reductions since reductions can be done easier in the communication complexity setting.

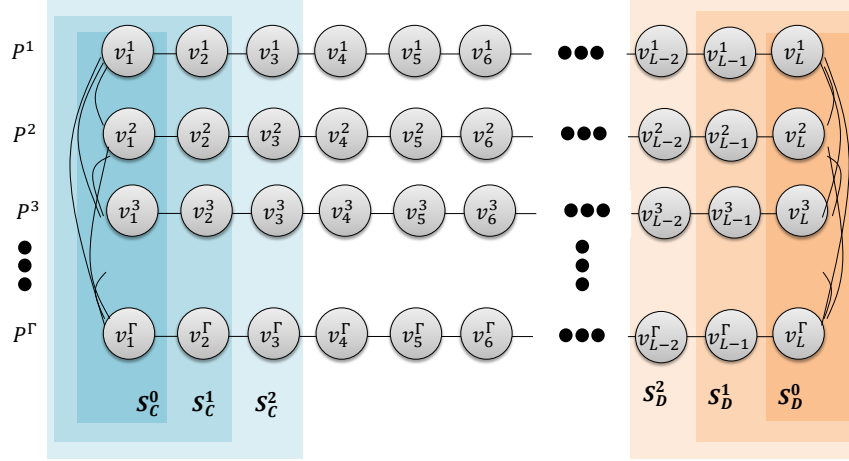


Figure 8: The network  $N'$  used in the proof idea of Theorem 10.1 with sets  $S_C^t$  and  $S_D^t$ .

To explain the main idea, let us focus on the non-gap version of Hamiltonian cycle verification and consider a  $B$ -model network  $N'$  consisting of  $\Gamma$  paths, each of length  $L$ , where we have an edge between any pair of the leftmost (respectively, rightmost) nodes of paths, as shown in Fig. 8. Now we will prove that if  $Q_{\epsilon_0, \epsilon_1}^{*, N}(\text{Ham}(N)) \leq (L/2) - 2$  then  $Q_{\epsilon_0, \epsilon_1}^{*, \text{server}}(\text{Ham}_\Gamma) = 0$  (i.e. no communication is needed from Carol and David to the server!). Note that this statement is much stronger than the theorem statement but it is not desirable since  $N'$  has diameter  $\Theta(L)$  which is too large. We will show how to modify  $N'$  to get the desired network  $N$  later.

Let paths in  $N'$  be  $P^1, \dots, P^\Gamma$  and nodes in path  $P^i$  be  $v_1^i, \dots, v_L^i$ . Let  $\mathcal{A}$  be an  $(\epsilon_0, \epsilon_1)$ -error quantum distributed algorithm that solves the Hamiltonian cycle verification problem on network  $N$  ( $\text{Ham}(N)$ ) in at most  $(L/2) - 2$  time.

We show that Carol, David and the server can solve the Hamiltonian cycle problem on a  $\Gamma$ -node input graph without any communication, essentially by “simulating”  $\mathcal{A}$  on some input subnetwork  $M$  corresponding to  $G = (U, E_C \cup E_D)$  in the following sense. When receiving  $E_C$  and  $E_D$ , the three parties will construct a subnetwork  $M$  of  $N'$  (without communication) in such a way that  $M$  is a Hamiltonian cycle if and only if  $G = (U, E_C \cup E_D)$  is. Next, they will simulate algorithm  $\mathcal{A}$  in such a way that, at any time  $t$  and for each node  $v_j^i$  in  $N'$ , there will be exactly one party among Carol, David and the server that knows *all information that  $v_j^i$  should know in order to run algorithm  $\mathcal{A}$* , i.e., the state of  $v_j^i$  as well as the messages (each consisting of  $B$  quantum bits) sent to  $v_j^i$  from its neighbors at time  $t$ . The party that knows this information will pretend to be  $v_j^i$  and apply algorithm  $\mathcal{A}$  to get the state of  $v_j^i$  at time  $t + 1$  as well as the messages that  $v_j^i$  will send to its neighbors at time  $t + 1$ . We say that this party *owns*  $v_j^i$  at time  $t$ . Details are as follows.

Initially at time  $t = 0$ , we let Carol owns all leftmost nodes, and David owns all rightmost nodes while the server owns the rest, i.e. Carol, David and the server own the following sets of nodes respectively (see Fig. 8):

$$S_C^0 = \{v_1^i \mid 1 \leq i \leq \Gamma\}, \quad S_D^0 = \{v_L^i \mid 1 \leq i \leq \Gamma\}, \quad S_S^0 = V(N') \setminus (S_C^0 \cup S_D^0). \quad (1)$$

After Carol and David each receive a perfect matching, denoted by  $E_C$  and  $E_D$  respectively, on the node set  $U = \{u_1, \dots, u_\Gamma\}$ , they construct a subnetwork  $M$  of  $N'$  as follows. For any  $i \neq j$ , Carol marks  $v_1^i v_1^j$  as participating in  $M$  if and only if  $u_i u_j \in E_C$ . Similarly, David marks  $v_L^i v_L^j$  as participating in  $M$  if and only if  $u_i u_j \in E_D$ . The server marks all edges in all paths as participating in  $M$ . Fig. 9 shows an example. We note the following observation which relies on the fact that  $E_C$  and  $E_D$  are perfect matchings.

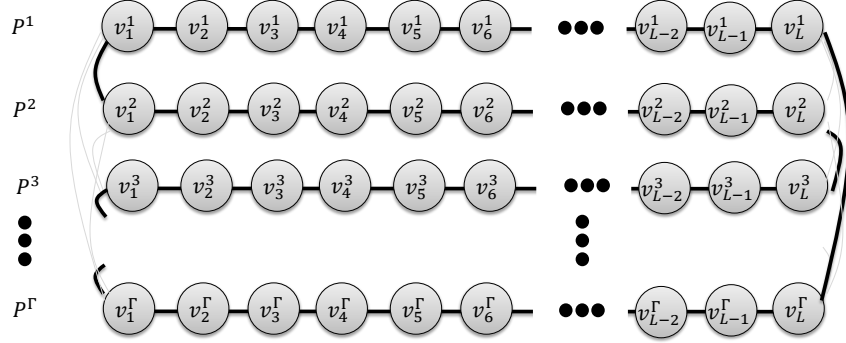


Figure 9: The subnetwork  $M$  when the input perfect matchings are  $E_C = \{(u_1, u_2), (u_3, u_4), \dots, (u_{\Gamma-1}, u_{\Gamma})\}$  and  $E_D = \{(u_2, u_3), (u_4, u_5), \dots, (u_{\Gamma}, u_1)\}$  (subnetwork  $M$  consists of all bold edges).

**Observation 10.2.** *The number of cycles in  $G = (U, E_C \cup E_D)$  is the same as the number of cycles in  $M$ .*

Now the three parties start a simulation. Recall that at time  $t = 0$  the three parties own nodes in the sets  $S_C^0$ ,  $S_D^0$  and  $S_S^0$  as in Eq.(1). Our goal is to simulate  $\mathcal{A}$  for one time step and make sure that Carol, David and the server own the following sets respectively (see Fig. 8):

$$S_C^1 = \{v_1^i, v_2^i \mid 1 \leq i \leq \Gamma\}, \quad S_D^1 = \{v_{L-1}^i, v_L^i \mid 1 \leq i \leq \Gamma\}, \quad S_S^1 = V(N') \setminus (S_C^1 \cup S_D^1). \quad (2)$$

To do this, the parties simulate  $\mathcal{A}$  on the nodes they own for one time step. This means that each of them will know the states and out-going messages at time  $t = 1$  (i.e., after  $\mathcal{A}$  is executed once) of nodes they own. Observe that although Carol knows the state of  $v_1^i$ , for any  $i$ , at time  $t = 1$ , she is not able to simulate  $\mathcal{A}$  on  $v_1^i$  for one more step since she does not know the message sent from  $v_2^i$  to  $v_1^i$  at time  $t = 1$ . This information is known by the server who owns  $v_2^i$  at time  $t = 0$ . Thus, we let the server send this message to Carol. Additionally, for Carol to own node  $v_2^i$  at time  $t = 1$ , it suffices to let the server send the state of  $v_2^i$  and the message sent from  $v_3^i$  to  $v_2^i$  at time  $t = 1$  (which are known by the server since it owns  $v_2^i$  and  $v_3^i$  at time  $t = 0$ ). The messages sent from the server to David can be constructed similarly. It can be checked that after this communication the three parties own nodes as in Eq.(2) and thus they can simulate  $\mathcal{A}$  for one more step.

Using a similar argument as the above we can guarantee that at any time  $t \leq (L/2) - 2$ , Carol, David and the server own nodes in the following sets respectively:

$$S_C^t = \{v_j^i \mid 1 \leq i \leq \Gamma, 1 \leq j \leq t+1\}, \quad S_D^t = \{v_j^i \mid 1 \leq i \leq \Gamma, L-t \leq j \leq L\}, \quad S_S^t = V(N') \setminus (S_C^t \cup S_D^t).$$

Thus, if algorithm  $\mathcal{A}$  terminates in  $(L/2) - 2$  steps then Carol, David and the server will know whether  $M$  is a Hamiltonian cycle or not with  $(\epsilon_0, \epsilon_1)$ -error by reading the output of nodes they own. By Observation 10.2, they will know whether  $G = (U, E_C \cup E_D)$  is a Hamiltonian cycle or not with the same error bound.

Now we modify  $N'$  to get network  $N$  of small diameter. A simple idea to slightly reduce the diameter is to add a path having half the number of nodes of other paths and connect its nodes to every other node on the other paths (see path  $H^1$  in Fig. 10). This path helps reducing the diameter from  $L$  to roughly  $(L/2) - 2$  since any pair of nodes can connect in roughly  $(L/2) - 2$  hops through this path. By adding about  $O(\log L)$  such paths (with  $H^i$  having half the number of nodes of  $H^{i-1}$ ) as in Fig. 10, we can reduce the diameter to  $O(\log L)$ . We call the new paths *highways*.

We can use almost the same argument as before to prove the theorem, by modifying sets  $S_C^t$ ,  $S_D^t$  and  $S_S^t$  appropriately as in Fig. 10 and consider the input graph  $G = (U, E_C \cup E_D)$  of  $\Gamma + k$  nodes,

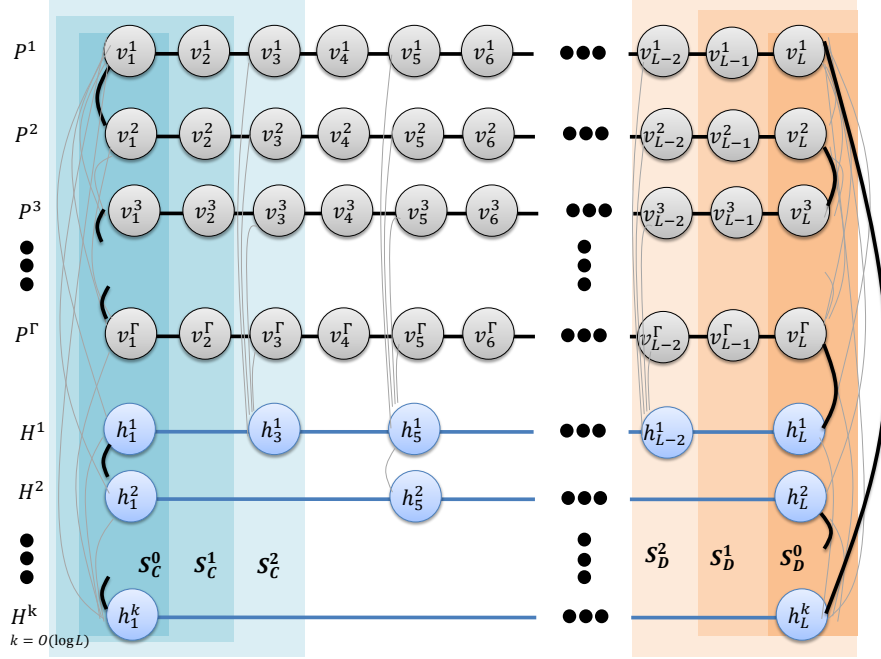


Figure 10: The network  $N$  which consists of network  $N'$  and some “highways” which are paths with nodes  $h_j^i$  (i.e., nodes in blue). Bold edges show an example of subnetwork  $M$  when the input perfect matchings are  $E_C = \{(u_1, u_2), (u_3, u_4), \dots, (u_{\Gamma+k-1}, u_{\Gamma+k})\}$  and  $E_D = \{(u_2, u_3), (u_4, u_5), \dots, (u_{\Gamma+k}, u_1)\}$ . Pale edges are those in  $N$  but not in  $M$ .

where  $k$  is the number of highways. The exception is that now Carol and David have to speak a little. For example, observe that if the three parties want to own the states of  $S_C^1$ ,  $S_D^1$  and  $S_S^1$  at time  $t = 1$ , Carol has to send to the server the messages sent from node  $h_1^i$  to its right neighbor, for all  $i$ . Since this message has size at most  $B$  and the simulation is done for  $Q_{\epsilon_0, \epsilon_1}^{*,N}(\text{Ham}(N))$  steps, Carol will send  $O((B \log n) Q_{\epsilon_0, \epsilon_1}^{*,N}(\text{Ham}(N)))$  qubits to the server. David will have to send the same amount of information and thus the complexity in the server model is as claimed.  $\square$

## 11 Proof of main theorems

### 11.1 Proof of Theorem 5.1

**Theorem 11.1** (Theorem 5.1 restated). *For any  $B$  and large  $n$ , there exists  $\epsilon > 0$  and a  $B$ -model  $n$ -node network  $N$  of diameter  $\Theta(\log n)$  such that any  $(\epsilon, \epsilon)$ -error quantum algorithm with prior entanglement for Hamiltonian cycle, and spanning tree verification problems on  $N$  requires  $\Omega(\sqrt{\frac{n}{B \log n}})$  time. That is,  $Q_{\epsilon, \epsilon}^{*,N}(\text{Ham}(N))$  and  $Q_{\epsilon, \epsilon}^{*,N}(\text{ST}(N))$  are  $\Omega(\sqrt{\frac{n}{B \log n}})$ .*

*Proof.* We note from Theorem 9.1 that

$$Q_{\epsilon, \epsilon}^{*,server}(\text{Ham}_\Gamma) > c'\Gamma \quad (3)$$

for some  $\epsilon > 0$  and  $c' > 0$ . Let  $c$  be the constant in the big-Oh in Theorem 10.1. Let  $L = \lfloor \frac{c'}{c} \sqrt{\frac{n}{B \log n}} \rfloor$  and  $\Gamma = \lceil \sqrt{Bn \log n} \rceil$ . Assume that

$$Q_{\epsilon, \epsilon}^{*,N}(\text{Ham}(N)) \leq L/2 \leq \frac{c'}{2c} \sqrt{\frac{n}{B \log n}}. \quad (4)$$

By Theorem 10.1, there is a network  $N$  of diameter  $O(\log L) = O(\log n)$  and  $\Theta(L\Gamma) = \Theta(n)$  nodes such that

$$\begin{aligned} Q_{\epsilon, \epsilon}^{*, server}(\text{Ham}_\Gamma) &\leq (cB \log L) Q_{\epsilon, \epsilon}^{*, N}(\text{Ham}(N)) \\ &\leq (cB \log L) \left( \frac{c'}{2c} \sqrt{\frac{n}{B \log n}} \right) \\ &\leq c' \sqrt{Bn \log n} \end{aligned}$$

where the second equality is by Eq. (4). This contradicts Eq.(3), thus proving that  $Q_{\epsilon, \epsilon}^{*, N}(\text{Ham}(N)) > L/2 \geq \frac{c'}{4c} \sqrt{\frac{n}{B \log n}}$ .

To show a lower bound of  $Q_{\epsilon, \epsilon}^{*, N}(\text{ST}(N))$ , let  $\mathcal{A}$  be an algorithm that solves spanning tree verification on  $N$  in  $T_{\mathcal{A}}$  time. We can use  $\mathcal{A}$  to verify if a subnetwork  $M$  is a Hamiltonian cycle as follows. First, we check that all nodes have degree two in  $M$  (this can be done in  $O(D)$  time). If not,  $M$  is not a Hamiltonian cycle. If it is then  $M$  consists of cycles. Now we delete one edge  $e$  in  $M$  arbitrarily, and use  $\mathcal{A}$  to check if this subnetwork is a spanning tree. It is easy to see that this subnetwork is a spanning tree if and only if  $M$  is a Hamiltonian cycle. The running time of our algorithm is  $T_{\mathcal{A}} + O(D)$ . The lower bound of  $Q_{\epsilon, \epsilon}^{*, N}(\text{Ham}(N))$  implies that  $T_{\mathcal{A}} = \Omega(\sqrt{\frac{n}{B \log n}})$ .  $\square$

### 11.2 Proof of Theorem 5.3

**Theorem 11.2** (Theorem 5.3 restated). *For any  $n, B, W$  and  $\alpha < W$  there exists  $\epsilon > 0$  and a  $B$ -model  $\Theta(n)$ -node network  $N$  of diameter  $\Theta(\log n)$  such that any  $\epsilon$ -error  $\alpha$ -approximation quantum algorithm with prior entanglement for computing the minimum spanning tree problem on  $N$  with weight function  $w : E(N) \rightarrow \mathbb{R}_+$  such that  $\frac{\max_{e \in E(N)} w(e)}{\min_{e \in E(N)} w(e)} \leq W$  requires  $\Omega(\frac{1}{\sqrt{B \log n}} \min(W/\alpha, \sqrt{n}))$  time.*

*Proof.* We note from Theorem 9.1 that

$$Q_{0, \epsilon}^{*, server}((\beta\Gamma)\text{-Ham}_\Gamma) > c'\Gamma \quad (5)$$

for some constant  $\beta > 0, \epsilon > 0$  and  $c' > 0$ . Let  $c$  be the constant in the big-Oh in Theorem 10.1. Let  $L = \lfloor \frac{c'}{c\sqrt{B \log n}} \min(\frac{W}{\alpha}, \sqrt{n}) \rfloor$  and  $\Gamma = \lceil \sqrt{B \log n} \max(\frac{n\alpha}{W}, \sqrt{n}) \rceil$ . We prove the following claim the same way we prove Theorem 5.1 in the previous section.

**Claim 11.3.**  $Q_{0, \epsilon}^{*, N}((\beta\Gamma)\text{-Ham}) > \frac{L}{2} \geq \frac{c'}{4c} \min(W/\alpha, \sqrt{\frac{n}{B \log n}})$

*Proof.* Assume that

$$Q_{0, \epsilon}^{*, N}((\beta\Gamma)\text{-Ham}) \leq \frac{L}{2} \leq \frac{c'}{2c} \min(W/\alpha, \sqrt{\frac{n}{B \log n}}). \quad (6)$$

By Theorem 10.1, there is a network  $N$  of diameter  $\Theta(\log L) = O(\log n)$  and  $\Theta(L\Gamma) = \Theta(n)$  nodes such that

$$\begin{aligned} Q_{0, \epsilon}^{*, server}((\beta\Gamma)\text{-Ham}_\Gamma) &\leq (cB \log L) Q_{0, \epsilon}^{*, N}((\beta\Gamma)\text{-Ham}) \\ &\leq (cB \log L)(L/2) \\ &\leq \frac{c' \sqrt{B \log n}}{2} \min(\frac{W}{\alpha}, \sqrt{n}) \\ &\leq \frac{c' \sqrt{B \log n}}{2} \max(\frac{n\alpha}{W}, \sqrt{n}) \\ &\leq c'\Gamma \end{aligned}$$

where the second equality is by Eq. (6) and the fourth inequality is because if  $\frac{W}{\alpha} \leq \sqrt{n}$  then  $\alpha \geq W/\sqrt{n}$  and thus  $n\alpha/W \geq \sqrt{n} \geq W/\alpha$ . This contradicts Eq.(5).  $\square$

Now assume that there is an  $\epsilon$ -error quantum distributed algorithm  $\mathcal{A}$  that finds an  $\alpha$ -approximate MST in  $T_{\mathcal{A}}$  time. We use  $\mathcal{A}$  to construct an  $(0, \epsilon)$ -error algorithm that solves  $(\beta\Gamma)$ -Ham( $N$ ) in  $T_{\mathcal{A}} + O(D)$  time as follows. Let  $M$  be the input subnetwork. First we check if all nodes have degree exactly two in  $M$ . If not then  $M$  is not a Hamiltonian cycle and we are done. If it is then  $M$  consist of one cycle or more. It is left to check whether  $M$  is connected or not. To do this, we assign weight 1 to all edges in  $H$  and weight  $W$  to the rest edges. We use  $\mathcal{A}$  to compute an  $\alpha$ -approximate MST  $T$ . Then we compute the weight of  $T$  in  $O(D) = O(\log n)$  rounds. If  $T$  has weight at most  $\alpha(n-1)$  then we say that  $H$  is connected; otherwise we say that it is  $(\beta\Gamma)$ -far from being connected.

To show that this algorithm is  $(0, \epsilon)$ -error, observe that, for any  $i$ , if  $H$  is  $i$ -far from being connected then the MST has weight at least  $(n-1-i) + iW$  since the MST will contain at least  $i$  edges of weight  $W$ . If  $H$  is connected then the MST has weight exactly  $n-1$  which means that  $T$  will have weight at most  $\alpha(n-1)$  with probability at least  $1-\epsilon$ , and we will say that  $H$  is connected with probability at least  $1-\epsilon$ . Otherwise, if  $H$  is  $(\beta\Gamma)$ -far from connected then  $T$  always have weight at least

$$(n-1-\beta\Gamma) + \beta\Gamma W \geq \beta\Gamma W \geq \beta(\sqrt{B \log n} \max(\frac{n\alpha}{W}, \sqrt{n}))W \geq \beta\sqrt{B \log n} \frac{n\alpha}{W} W \geq \alpha n > \alpha(n-1)$$

for large enough  $n$  (note that  $\beta$  is a constant), and we will always say that  $H$  is  $(\beta\Gamma)$ -far from being connected. Thus algorithm is  $(0, \epsilon)$ -error as claimed.  $\square$

# Part II: Formal Definitions and Further Technical Details

## 12 Detailed Definitions

### 12.1 Quantum Distributed Network Models

#### Informal descriptions

We first describe a *general* model which will later make it easier to define some specific models we are considering. We assume some familiarity with quantum computation (see, e.g., [51, 67] for excellent resources). A general distributed network  $N$  is modeled by a set of  $n$  processors, denoted by  $u_1, \dots, u_n$ , and a set of *bandwidth* parameters between each pair of processors, denoted by  $B_{u_i u_j}$  for any  $i \neq j$ , which is used to bound the size of messages sent from  $u_i$  to  $u_j$ . Note that  $B_{u_i u_j}$  could be zero or infinity. To simplify our formal definition, we let  $B_{u_i u_i} = \infty$  for all  $i$ .

In the beginning of the computation, each processor  $u_i$  receives an input string  $x_i$ , each of size  $b$ . The processors want to cooperatively compute a global function  $f(x_1, \dots, x_n)$ . They can do this by communicating in *rounds*. In each rounds, processor  $u_i$  can send a message of  $B_{u_i u_j}$  bits or qubits to processor  $u_j$ . (Note that  $u_i$  can send different messages to  $u_j$  and  $u_k$  for any  $j \neq k$ .) We assume that each processor has unbounded computational power. Thus, between each round of communication, processors can perform any computation (even solving an NP-complete problem!). The *time complexity* is the minimum number of rounds needed to compute the function  $f$ . We can categorize this model further based on the type of communication (classical or quantum) and computation (deterministic or randomized).

In this paper, we are interested in quantum communication when errors are allowed and nodes share entangled qubits. In particular, for any  $\epsilon > 0$  and function  $f$ , we say that a quantum distributed algorithm  $\mathcal{A}$  is  $\epsilon$ -error if for any input  $(x_1, \dots, x_n)$ , after  $\mathcal{A}$  is executed on this input any node  $u_i$  knows the value of  $f(x_1, \dots, x_n)$  correctly with probability at least  $1 - \epsilon$ . We let  $Q_\epsilon^{*,N}(N)$  denote the time complexity (number of rounds) of computing function  $f$  on network  $N$  with  $\epsilon$ -error.

In the special case where  $f$  is a boolean function, for any  $\epsilon_0, \epsilon_1 > 0$  we say that  $\mathcal{A}$  computes  $f$  with  $(\epsilon_0, \epsilon_1)$ -error if, after  $\mathcal{A}$  is executed on any input  $(x_1, \dots, x_n)$ , any node  $u_i$  knows the value of  $f(x_1, \dots, x_n)$  correctly with probability at least  $1 - \epsilon_0$  if  $f(x_1, \dots, x_n) = 0$  and with probability at least  $1 - \epsilon_1$  otherwise. We let  $Q_{\epsilon_0, \epsilon_1}^{*,N}(N)$  denote the time complexity of computing boolean function  $f$  on network  $N$  with  $(\epsilon_0, \epsilon_1)$ -error.

Two main models of interest are the  $B$ -model (also known as  $\text{CONGEST}(B)$ ) and a new model we introduce in this paper called the *server model*. The  $B$ -model is modeled by an undirected  $n$ -node graph, where vertices model the processors and edges model the links between the processors. For any nodes (processors)  $u_i$  and  $u_j$ ,  $B_{u_i u_j} = B_{u_j u_i} = B$  if there is an edge  $u_i u_j$  in the graph and  $B_{u_i u_j} = B_{u_j u_i} = 0$  otherwise.

In the server model, there are three processors, denoted by *Carol*, *David* and the *server*. In each round, Carol and David can send one bit to each other and to the server while receiving an arbitrarily large message from the server, i.e.  $B_{\text{Carol}, \text{David}} = B_{\text{David}, \text{Carol}} = B_{\text{Carol}, \text{Server}} = B_{\text{David}, \text{Server}} = 1$  and  $B_{\text{Server}, \text{Carol}} = B_{\text{Server}, \text{David}} = \infty$ .

We will also discuss the *two-party communication complexity model* which is simply the network of two processors called Alice and Bob with bandwidth parameters  $B_{\text{Alice}, \text{Bob}} = B_{\text{Bob}, \text{Alice}} = 1$ . (Note that, this model is sometimes defined in such a way that only one of the processors can send a message in each round. The communication complexity in this setting might be different from ours, but only by a factor of two.)

When  $N$  is the server or two-party communication complexity model, we use  $Q_\epsilon^{*,\text{server}}(f)$  and  $Q_\epsilon^{*,cc}(f)$  instead of  $Q_\epsilon^{*,N}(f)$ .

## Formal definitions

**Network States** The *pure state* of a quantum network of  $n$  nodes with parameters  $\{B_{u_i u_j}\}_{1 \leq i, j \leq n}$  is represented as a vector in a Hilbert space

$$\bigotimes_{1 \leq i, j \leq n} H_{u_i u_j} = H_{u_1 u_1} \otimes H_{u_1 u_2} \otimes \dots \otimes H_{u_1 u_n} \otimes H_{u_2 u_1} \otimes \dots \otimes H_{u_2 u_n} \otimes \dots \otimes H_{u_n u_n}$$

where  $\otimes$  is the tensor product. Here,  $H_{u_i u_i}$ , for any  $i$ , is a Hilbert space of arbitrary finite dimension representing the “workspace” of processor  $u_i$ . In particular, we let  $K$  be an arbitrarily large number (thus the complexity of the problem cannot depend on  $K$ ) and  $H_{u_i u_i}$  be a  $2^K$ -dimensional Hilbert space. Additionally,  $H_{u_i u_j}$ , for any  $i \neq j$ , is a Hilbert space representing the  $B_{u_i u_j}$ -qubit communication channel from  $u_i$  to  $u_j$ . Its dimension is  $2^{B_{u_i u_j}}$  if  $B_{u_i u_j}$  is finite and  $2^K$  if  $B_{u_i u_j} = \infty$ .

The *mixed state* of a quantum network  $N$  is a probabilistic distribution over its pure states

$$\{(p_i, |\psi_i\rangle)\} \text{ with } p_i \geq 0 \text{ and } \sum_i p_i = 1.$$

We note that it is sometimes convenient to represent a mixed state by a *density matrix*  $\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i|$ .

**Initial state** In the model *without* prior entanglement, the initial (pure) state of a quantum protocol on input  $(x_1, \dots, x_n)$  is the vector

$$|\psi_{x_1, \dots, x_n}^0\rangle = \bigotimes_{1 \leq i, j \leq n} |\psi_{x_1, \dots, x_n}^0(i, j)\rangle = |\psi_{x_1, \dots, x_n}^0(1, 1)\rangle |\psi_{x_1, \dots, x_n}^0(1, 2)\rangle \dots |\psi_{x_1, \dots, x_n}^0(n, n)\rangle$$

where  $|\psi_{x_1, \dots, x_n}^0(i, j)\rangle$  for any  $1 \leq i, j \leq n$  is a vector in  $H_{u_i u_j}$  such that  $|\psi_{x_1, \dots, x_n}^0(i, i)\rangle = |x_i, 0\rangle$  for any  $i$  and  $|\psi_{x_1, \dots, x_n}^0(i, j)\rangle = |0\rangle$  for any  $i \neq j$  (here,  $|0\rangle$  represents an arbitrary unit vector independent of the input). Informally, this corresponds to the case where each processor  $u_i$  receives an input  $x_i$  and workspaces and communication channel are initially “clear”.

With prior entanglement, the initial (pure) state is a unit vector of the form

$$|\psi_{x_1, \dots, x_n}^0\rangle = \sum_w \left( \alpha_w \bigotimes_{1 \leq i, j \leq n} |\psi_{w, x_1, \dots, x_n}^0(i, j)\rangle \right) \quad (7)$$

where  $|\psi_{w, x_1, \dots, x_n}^0(i, j)\rangle$  for any  $1 \leq i, j \leq n$  is a vector in  $H_{u_i u_j}$  such that  $|\psi_{w, x_1, \dots, x_n}^0(i, i)\rangle = |x_i, w\rangle$  for any  $i$  and  $|\psi_{w, x_1, \dots, x_n}^0(i, j)\rangle = |0\rangle$  for any  $i \neq j$ . Here, the coefficients  $\alpha_w$  are arbitrary real numbers satisfying  $\sum_w \alpha_w^2 = 1$  that is independent of the input  $(x_1, \dots, x_n)$ . Informally, this corresponds to the case where processors share entangled qubits in their workspaces.

Note that we can assume the global state of the network to be always a pure state, since any mixed state can be purified by adding qubits to the processor’s workspaces, and ignoring these in later computations.

**Communication Protocol** The communication protocol consists of rounds of *internal computation* and *communication*. In each internal computation of the  $t^{\text{th}}$  round, each processor  $u_i$  applies a *unitary transformation* to its incoming communication channels and its own memory, i.e.  $H_{u_j u_i}$  for all  $j$ . That is, it applies a unitary transformation of the form

$$C_{t, u_i} \otimes \left( \bigotimes_{1 \leq j \leq n, k \neq i} I_{u_j u_k} \right) \quad (8)$$



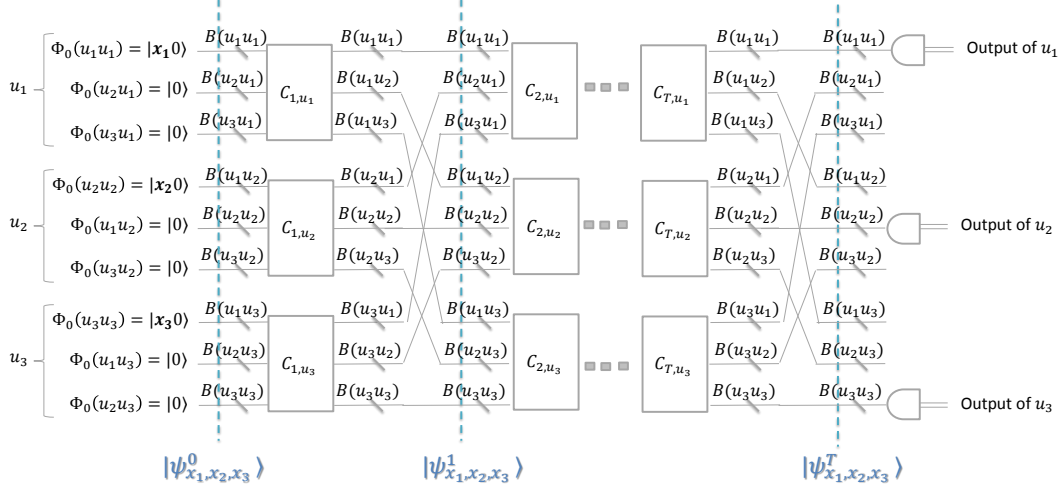


Figure 11: A circuit corresponding to  $T$  rounds of communication on general distributed network having 3 processors. The information flows from left to right and the line crossing each wire with a number  $B_{u_i u_j}$  means that there are  $B_{u_i u_j}$  qubits of information flowing through such wire. We note that the initial state in the picture is without entanglement.

which acts as an identity on  $H_{u_j u_k}$  for all  $1 \leq j \leq n$  and  $k \neq i$ . At the end of the internal computation, we require the communication channel to be clear, i.e. if we would measure any communication channel in the computational basis then we would get  $|0\rangle$  with probability one. This can easily be achieved by swapping some fresh qubits from the private workspace into the communication channel. Note that the processors can apply the transformations corresponding to an internal computation simultaneously since they act on different parts of the network's state.

To define communication, let us divide the workspace  $H_{u_i u_i}$  of processor  $u_i$  further to

$$H_{u_i u_i} = H_{u_i u_i,1} \otimes H_{u_i u_i,2} \otimes \dots \otimes H_{u_i u_i,n}$$

where  $H_{u_i u_i,j}$  has the same dimension as  $H_{u_i u_j}$ . The space  $H_{u_i u_i,j}$  can be thought of as a place where  $u_i$  prepares the messages it wants to send to  $u_j$  in each round, while  $H_{u_i u_i,i}$  holds  $u_i$ 's remaining workspace. Now, for any  $j \neq i$ ,  $u_i$  sends a message to  $u_j$  simply by swapping the qubits in  $H_{u_i u_i,j}$  with those in  $H_{u_i u_j}$ . Note that  $u_i$  does not receive any information in this process since the communication channel  $H_{u_i u_j}$  is clear after the internal computation. Also note that we can perform the swapping operations between any pair  $i \neq j$  simultaneously since they act on different part of the network state. This completes one round of communication. We let

$$|\psi_{x_1, \dots, x_n}^t\rangle \quad (9)$$

denote the network state after  $t$  rounds of communication.

At the end of a  $T$ -round protocol, we compute the *output of processor  $u_i$*  as follows. We view part of  $H_{u_i u_i}$  as an output space of  $u_i$ , i.e.  $H_{u_i u_i} = H_{O_i} \otimes H_{W_i}$  for some  $H_{O_i}$  and  $H_{W_i}$ . We compute the output of  $u_i$  by measuring  $H_{O_i}$  in the computational basis. That is, if we let  $K'$  be the number of qubits in  $H_{O_i}$  and the network state after a  $T$ -round protocol be  $\psi_{x_1, \dots, x_n}^T$  then, for any  $w \in \{0, 1\}^{K'}$ ,

$$\Pr[\text{Processor } u_i \text{ outputs } w] = |\langle \psi_{x_1, \dots, x_n}^T | w \rangle|^2.$$

Fig. 11 depicts a quantum circuit corresponding to a communication protocol on three processors.

**Error and Time Complexity** For any  $0 \leq \epsilon \leq 1$ , we say that a quantum protocol  $\mathcal{A}$  on network  $N$  computes function  $f$  with  $\epsilon$ -error if for any input  $(x_1, \dots, x_n)$  of  $f$  and any processor  $u_i$ ,  $u_i$  outputs  $f(x_1, \dots, x_n)$  with probability at least  $1 - \epsilon$  after  $\mathcal{A}$  is executed. The  $\epsilon$ -error time complexity of computing function  $f$  on network  $N$ , denoted by  $Q_\epsilon^{*,N}(f)$ , is the minimum  $T$  such that there exists a  $T$ -round quantum protocol on network  $N$  that computes function  $f$  with  $\epsilon$ -error. We note that we allow the protocol to start with an entangled state. The  $*$  in the notation follows the convention to contrast with the case that we do not allow prior entanglement (which is not considered in this paper). When  $N$  is the server model and two-party communication complexity model mentioned earlier, we use  $Q_\epsilon^{*,server}(f)$  and  $Q_\epsilon^{*,cc}(f)$  respectively to denote the  $\epsilon$ -error time complexity.

If  $f$  is a boolean function, we will sometimes distinguish between the error of outputting 0 and 1. For any  $0 \leq \epsilon_0, \epsilon_1 \leq 1$  we say that  $\mathcal{A}$  computes  $f$  with  $(\epsilon_0, \epsilon_1)$ -error if for any input  $(x_1, \dots, x_n)$  of  $f$  and any processor  $u_i$ , if  $f(x_1, \dots, x_n) = 0$  then  $u_i$  outputs 0 with probability at least  $1 - \epsilon_0$  and otherwise  $u_i$  outputs 1 with probability at least  $1 - \epsilon_1$ . The time complexity, denoted by  $Q_{\epsilon_0, \epsilon_1}^{*,N}(f)$  is defined in the same way as before. We will also use  $Q_{\epsilon_0, \epsilon_1}^{*,server}(f)$  and  $Q_{\epsilon_0, \epsilon_1}^{*,cc}(f)$ .

## 12.2 Distributed Graph Verification Problems

In the distributed network  $N$ , we describe its subgraph  $M$  as an input as follows. Each node  $u_i$  in  $N$  receives an  $n$ -bit binary string  $x_{u_i}$  as an input. We let  $x_{u_i, u_1}, \dots, x_{u_i, u_n}$  be the bits of  $x_{u_i}$ . Each bit  $x_{u_i, u_j}$  indicates whether edge  $u_i v_j$  participates in the subgraph  $M$  or not. The indicator variables must be consistent, i.e., for every edge  $u_i u_j \in E(N)$ ,  $x_{u_i, u_j} = x_{u_j, u_i}$  (this is easy to verify with a single round of communication) and if there is no edge between  $u_i$  and  $u_j$  in  $N$  then  $x_{u_i, u_j} = x_{u_j, u_i} = 0$ .

We define  $M_{x_{u_1}, \dots, x_{u_n}}$ , or simply  $M$ , to be subgraph of  $N$  having edges whose indicator variables are 1; that is,

$$E(M) = \{(u_i, u_j) \in E \mid \forall i \neq j, x_{u_i, u_j} = x_{u_j, u_i} = 1\}.$$

We list the following problems concerning the verification of properties of subnetwork  $M$  on distributed network  $N$  from [14].

- **connected spanning subgraph verification:** We want to verify whether  $M$  is connected and spans all nodes of  $N$ , i.e., every node in  $N$  is incident to some edge in  $M$ .
- **cycle containment verification:** We want to verify if  $M$  contains a cycle.
- **$e$ -cycle containment verification:** Given an edge  $e$  in  $M$  (known to vertices adjacent to it), we want to verify if  $M$  contains a cycle containing  $e$ .
- **bipartiteness verification:** We want to verify whether  $M$  is bipartite.
- **$s$ - $t$  connectivity verification:** In addition to  $N$  and  $M$ , we are given two vertices  $s$  and  $t$  ( $s$  and  $t$  are known by every vertex). We would like to verify whether  $s$  and  $t$  are in the same connected component of  $M$ .
- **connectivity verification:** We want to verify whether  $M$  is connected.
- **cut verification:** We want to verify whether  $M$  is a cut of  $N$ , i.e.,  $N$  is not connected when we remove edges in  $M$ .
- **edge on all paths verification:** Given two nodes  $u, v$  and an edge  $e$ . We want to verify whether  $e$  lies on all paths between  $u$  and  $v$  in  $M$ . In other words,  $e$  is a  $u$ - $v$  cut in  $M$ .
- **$s$ - $t$  cut verification:** We want to verify whether  $M$  is an  $s$ - $t$  cut, i.e., when we remove all edges  $E(M)$  of  $M$  from  $N$ , we want to know whether  $s$  and  $t$  are in the same connected component or not.

- **least-element list verification [13, 30]:** The input of this problem is different from other problems and is as follows. Given a distinct rank (integer)  $r(v)$  to each node  $v$  in the weighted graph  $N$ , for any nodes  $u$  and  $v$ , we say that  $v$  is the *least element* of  $u$  if  $v$  has the lowest rank among vertices of distance at most  $d(u, v)$  from  $u$ . Here,  $d(u, v)$  denotes the weighted distance between  $u$  and  $v$ . The *Least-Element List* (LE-list) of a node  $u$  is the set  $\{\langle v, d(u, v) \rangle \mid v \text{ is the least element of } u\}$ .

In the least-element list verification problem, each vertex knows its rank as an input, and some vertex  $u$  is given a set  $S = \{\langle v_1, d(u, v_1) \rangle, \langle v_2, d(u, v_2) \rangle, \dots\}$  as an input. We want to verify whether  $S$  is the least-element list of  $u$ .

- **Hamiltonian cycle verification:** We would like to verify whether  $M$  is a Hamiltonian cycle of  $N$ , i.e.,  $M$  is a simple cycle of length  $n$ .
- **spanning tree verification:** We would like to verify whether  $M$  is a tree spanning  $N$ .
- **simple path verification:** We would like to verify that  $M$  is a simple path, i.e., all nodes have degree either zero or two in  $M$  except two nodes that have degree one and there is no cycle in  $M$ .

### 12.3 Distributed Graph Optimization Problems

In the graph optimization problems  $\mathcal{P}$  on distributed networks, such as finding MST, we are given a positive weight  $\omega(e)$  on each edge  $e$  of the network (each node knows the weights of all edges incident to it). Each pair of network and weight function  $(N, \omega)$  comes with a nonempty set of *feasible solution* for problem  $\mathcal{P}$ ; e.g., for the case of finding MST, all spanning trees of  $N$  are feasible solutions. The goal of  $\mathcal{P}$  is to find a feasible solution that minimizes or maximizes the total weight. We call such solution an *optimal solution*. For example, the spanning tree of minimum weight is the optimal solution for the MST problem. We let  $W = \max_{e \in E(N)} \omega(e) / \min_{e \in E(N)} \omega(e)$ .

For any  $\alpha \geq 1$ , an  $\alpha$ -*approximate solution* of  $\mathcal{P}$  on weighted network  $(N, \omega)$  is a feasible solution whose weight is not more than  $\alpha$  (respectively,  $1/\alpha$ ) times of the weight of the optimal solution of  $\mathcal{P}$  if  $\mathcal{P}$  is a minimization (respectively, maximization) problem. We say that an algorithm  $\mathcal{A}$  is an  $\alpha$ -approximation algorithm for problem  $\mathcal{P}$  if it outputs an  $\alpha$ -approximate solution for any weighted network  $(N, \omega)$ . In case we allow errors, we say that an  $\alpha$ -approximation  $T$ -time algorithm is  $\epsilon$ -error if it outputs an answer that is not  $\alpha$ -approximate with probability at most  $\epsilon$  and always finishes in time  $T$ , regardless of the input.

Note the following optimization problems on distributed network  $N$  from [14].

- In the **minimum spanning tree** problem [22, 55], we want to compute the weight of the minimum spanning tree (i.e., the spanning tree of minimum weight). In the end of the process all nodes should know this weight.
- Consider a network with two cost functions associated to edges, weight and length, and a root node  $r$ . For any spanning tree  $T$ , the radius of  $T$  is the maximum length (defined by the length function) between  $r$  and any leaf node of  $T$ . Given a root node  $r$  and the desired radius  $\ell$ , a **shallow-light tree** [54] is the spanning tree whose radius is at most  $\ell$  and the total weight is minimized (among trees of the desired radius).
- Given a node  $s$ , the  **$s$ -source distance** problem [21] is to find the distance from  $s$  to every node. In the end of the process, every node knows its distance from  $s$ .

- In the **shortest path tree** problem [22], we want to find the shortest path spanning tree rooted at some input node  $s$ , i.e., the shortest path from  $s$  to any node  $t$  must have the same weight as the unique path from  $s$  to  $t$  in the solution tree. In the end of the process, each node should know which edges incident to it are in the shortest path tree.
- The **minimum routing cost spanning tree** problem [68] is defined as follows. We think of the weight of an edge as the cost of routing messages through this edge. The routing cost between any node  $u$  and  $v$  in a given spanning tree  $T$ , denoted by  $c_T(u, v)$ , is the distance between them in  $T$ . The routing cost of the tree  $T$  itself is the sum over all pairs of vertices of the routing cost for the pair in the tree, i.e.,  $\sum_{u,v \in V(N)} c_T(u, v)$ . Our goal is to find a spanning tree with minimum routing cost.
- A set of edges  $E'$  is a **cut** of  $N$  if  $N$  is not connected when we delete  $E'$ . The **minimum cut** problem [20] is to find a cut of minimum weight. A set of edges  $E'$  is an  $s$ - $t$  **cut** if there is no path between  $s$  and  $t$  when we delete  $E'$  from  $N$ . The **minimum  $s$ - $t$  cut** problem is to find an  $s$ - $t$  cut of minimum weight.
- Given two nodes  $s$  and  $t$ , the **shortest  $s$ - $t$  path** problem is to find the length of the shortest path between  $s$  and  $t$ .
- The **generalized Steiner forest** problem [30] is defined as follows. We are given  $k$  disjoint subsets of vertices  $V_1, \dots, V_k$  (each node knows which subset it is in). The goal is to find a minimum weight subgraph in which each pair of vertices belonging to the same subsets is connected. In the end of the process, each node knows which edges incident to it are in the solution.

## 13 Detailed Proofs of Section 8

### 13.1 Two-player XOR Games

We give a brief description of XOR games. AND game can be described similarly (their formal description is not needed in this paper). For a more detailed description as well as the more general case of nonlocal games see, e.g., [42, 6] and references therein. An XOR game is played by three parties, Alice, Bob and a referee. The game is defined by  $\mathcal{X}$  and  $\mathcal{Y}$  which is the set of input to Alice and Bob, respectively,  $\pi$ , a joint probability distribution  $\pi : \mathcal{X} \times \mathcal{Y} \rightarrow [0, 1]$ , and a boolean function  $f : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$ .

At the start of the game, the referee picks a pair  $(x, y) \in \mathcal{X} \times \mathcal{Y}$  according to the probability distribution  $\pi$  and sends  $x$  to Alice and  $y$  to Bob. Alice and Bob then answer the referee with one-bit message  $a$  and  $b$ . The players win the game if the value  $a \oplus b$  is equal to  $f(x, y)$ . In other words, Alice and Bob want the XOR of their answers to agree with  $f$ , explaining the name “XOR game.”

The goal of the players is to maximize the *bias* of the game, denoted by  $\text{Bias}_\pi(f)$ , which is the probability that Alice and Bob win minus the probability that they lose. In the classical setting, this is

$$\begin{aligned} \text{Bias}_\pi(f) &= \max_{\substack{a: \mathcal{X} \rightarrow \{-1, 1\}, \\ b: \mathcal{Y} \rightarrow \{-1, 1\}}} \sum_{(x, y) \in \mathcal{X} \times \mathcal{Y}} (-1)^{f(x, y)} \pi(x, y) (-1)^{a(x)} (-1)^{b(y)} \\ &= \max_{\substack{a \in \{-1, 1\}^{|\mathcal{X}|}, \\ b \in \{-1, 1\}^{|\mathcal{Y}|}}} \mathbb{E}_{(x, y) \sim \pi} [(-1)^{a(x)} (-1)^{b(y)} (-1)^{f(x, y)}]. \end{aligned}$$

In the quantum setting, Alice and Bob are allowed to play an *entangled strategy* where they may make use of an entangled state they share prior to receiving the input. That is, Alice and Bob start

with some shared pure quantum state which is independent of the input and after they receive input  $(x, y)$  they make some projective measurements depending on  $(x, y)$  and return the result of their measurements to the referee. Formally, an XOR entangled strategy is described by a shared (pure) quantum state  $|\psi\rangle \in \mathbb{C}^{d \times d}$  for some  $d \geq 1$  and a choice of projective measurements  $\{A_x^0, A_x^1\}$  and  $\{B_y^0, B_y^1\}$  for all  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$ . When receiving input  $x$  and  $y$ , the probability that Alice and Bob output  $(a, b) \in \{0, 1\}^2$  is  $\langle \psi | A_x^a \otimes B_y^b | \psi \rangle$ . Thus, the maximum correlation can be shown to be (see [6] for details)

$$\text{Bias}_\pi(f) = \max \mathbb{E}_{(x,y) \sim \pi} [\langle \psi | (A_x^1 - A_x^0) \otimes (B_y^1 - B_y^0) | \psi \rangle (-1)^{f(x,y)}]$$

where the maximization is over pure states  $|\psi\rangle$  and projective measurements  $\{A_x^0, A_x^1\}$  and  $\{B_y^0, B_y^1\}$  for all  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$ . In the rest of this paper,  $\text{Bias}_\pi(f)$  always denotes the maximum correlation in the quantum setting. We let

$$Q^{*, \text{XOR}}(f) = \min_{\pi} \text{Bias}_\pi(f).$$

We note that while the players could start the game with a mixed state, it can be shown that pure entangled states suffice in order to maximize the winning probability (see, e.g., [6]).

### 13.2 From Nonlocal Games to Server-Model Lower Bounds

**Lemma 13.1** (Lemma 8.2 restated). *For any boolean function  $f$  and  $\epsilon_0, \epsilon_1 \geq 0$ , there is an XOR-game strategy  $\mathcal{A}'$  and AND-game strategy  $\mathcal{A}''$  such that, for any input  $(x, y)$ ,*

- *with probability  $4^{-2Q_{\epsilon_0, \epsilon_1}^{*, \text{server}}(f)}$ ,  $\mathcal{A}'$  and  $\mathcal{A}''$  are able to simulate a protocol in the server model and hence output  $f(x, y)$  with probability at least  $1 - \epsilon_{f(x,y)}$ ;*
- *otherwise  $\mathcal{A}'$  outputs 0 and 1 with probability  $1/2$  each, and  $\mathcal{A}''$  outputs 0 with probability 1.*

*Proof.* We have sketched the proof in Section 8.1. We now provide more detail.

Let  $c = Q_{\epsilon_0, \epsilon_1}^{*, \text{server}}(f)$ , i.e. Carol and David communicate with the server for  $c$  rounds where each of them sends one qubit to the server per round while the server sends them messages of arbitrary size. While Alice and Bob cannot run a protocol  $\mathcal{A}$  in the server model since they cannot communicate to each other, we show that they can obtain the output of  $\mathcal{A}$  with probability  $\frac{1}{4^{2c}}$ . To be precise, for any input  $(x, y)$  let  $p_{x,y}$  and  $q_{x,y}$  be the probability that  $\mathcal{A}(x, y)$  is zero and one respectively. We will show that

$$\begin{aligned} &\text{Alice and Bob can obtain the final state of } \mathcal{A} \text{ with probability } 4^{-2c} \text{ and in that case} \\ &\text{output the correct answer with high probability. If they do not obtain that state one} \\ &\text{of them will output a random bit for XOR games and one of them will output 0 for} \\ &\text{AND games.} \end{aligned} \tag{10}$$

Hence the XOR game will accept with probability  $\frac{1}{2}(1 - 4^{-2c}) + 4^{-2c}q_{x,y} = \frac{1}{2} + (q_{x,y} - \frac{1}{2})4^{-2c}$  and thus have a bias of at least  $4^{-2c} \cdot \min\{1/2 - \epsilon_0, 1/2 - \epsilon_1\}$ .

The AND game will accept 1-inputs with probability at least  $q'_{x,y} \geq \frac{q_{x,y}}{4^{2c}}$ . Furthermore if  $\mathcal{A}$  never accepts a 0-input, then neither will the AND game.

Let us first prove Statement (10) with an additional assumption that there is a “fake” server that Alice and Bob can receive a message from but cannot talk to (we will eliminate this fake server later). We will call this a fake server to distinguish it from the “real” server in the server model.

First let us note the Carol and David need not talk to each other, but can send their messages to the server who can pass them to the other player. Since the server can also set up entanglement between the three parties without cost, Carol, David and the server can use *teleportation* (see [51])

for details) and we can assume that in protocol  $\mathcal{A}$  Carol and David send 2 classical bits per round to the server instead of one qubit. These two bits are also uniformly distributed, regardless of the state of the qubit.

Thus, for any input  $(x, y)$ , the messages sent by Carol and David in protocol  $\mathcal{A}$  will be  $a, b \in \{0, 1\}^{2c}$  with some probability, say  $p_{x,y,a,b}$ . For simplicity, let us assume that each communication sequence  $(a, b)$  leads to a unique output of  $\mathcal{A}$  on input  $(x, y)$  (e.g., by requiring Carol and David to send their result to the server in the last round). Let  $\mathcal{A}(x, y, a, b)$  be the output of the protocol  $\mathcal{A}$  on input  $(x, y)$  with communication sequence  $(a, b)$ . Then the probability that  $\mathcal{A}$  outputs zero and one is, respectively,

$$p_{x,y} = \sum_{(a,b): \mathcal{A}(x,y,a,b)=0} p_{x,y,a,b} \quad \text{and} \quad q_{x,y} = \sum_{(a,b): \mathcal{A}(x,y,a,b)=1} p_{x,y,a,b}.$$

The strategy of Alice and Bob who play the XOR and AND games is trying to “guess” this sequence.

In particular, Alice, Bob and the fake server will pretend to be Carol, David and the real server as follows. Before receiving the input, Alice, Bob and the fake server use their shared entanglement to create two shared random strings of length  $2c$ , denoted by  $a'$  and  $b'$ , and start their initial entangled states with the same states of Carol, David and the server. In each round  $t$  of  $\mathcal{A}$ , Alice, Bob and the fake server will simulate Carol, David and the real server, respectively, as follows. Let  $c_{t,1}$  and  $c_{t,2}$  be two bits sent by Carol to the real server at round  $t$ . Alice will check whether the guessed communication sequence  $a'$  is correct by checking if  $c_{t,1}$  and  $c_{t,2}$  are the same as  $a'_{2t-1}$  and  $a'_{2t}$  which are the  $(2t-1)^{th}$  and  $(2t)^{th}$  bits of  $a'$ . If they are not the same then she will ‘abort’ which means that

- Alice will output 0 and 1 uniformly random if she is playing an XOR game, and
- Alice will output 0 if she is playing an AND game.

Similarly, Bob will check whether the guessed communication sequence  $b'$  is correct by checking  $b'_{2t-1}$  and  $b'_{2t}$  with two classical bits sent by David to the server. Moreover, the fake server will pretend that it receives  $a'_{2t-1}$ ,  $a'_{2t}$ ,  $b'_{2t-1}$  and  $b'_{2t}$  to execute  $\mathcal{A}$  and send huge quantum messages to Alice and Bob. Alice and Bob then execute  $\mathcal{A}$  using these messages. After  $2c$  rounds (if no player aborts), the players output the following.

- In XOR games, Alice will send Carol’s output to the referee, and Bob will send 0 to the referee.
- In AND games, Alice will send Carol’s output to the referee, and Bob will send 1 to the referee.

Thus, if one or both players aborts then the output of an XOR game will be uniformly random in  $\{0, 1\}$ . For an AND game in case of a abort the players reject. Otherwise, the result of the XOR and AND games will be  $\mathcal{A}(x, y, a, b)$ . The probability that Alice and Bob do not abort, given that the communication sequence of  $\mathcal{A}$  on input  $(x, y)$  is  $a$  and  $b$  is  $Pr[a' = a \wedge b' = b] = \frac{1}{4^{2c}}$ .

This almost proves Statement (10) (thus the lemma) except that there is a fake server sending information to Alice and Bob in the XOR and AND game strategy. To remove the fake server, observe that we do not need an input in order to generate the messages the fake server sent to Alice and Bob. Thus, we change the strategy to the following. As previously done, before Alice, Bob and the fake server receive an input they generate shared random strings  $(a', b')$  and start with the initial states of Carol, David and the real server. In addition to this, the fake server use the string  $a'$  and  $b'$  to generate the messages sent by the real server to Carol and David. It then sends this information to Alice and Bob. We now remove the fake server completely and mark this point as a starting point of the XOR and AND games. After Alice and Bob receive input  $(x, y)$ , they simulate

protocol  $\mathcal{A}$  as before. In each round, when they are supposed to receive messages from the fake server, they read messages that the fake server sent before the game starts. Since the fake server sends the same messages, regardless of when it sends, the result is the same as before. Thus, we achieve Statement (10) even when there is no fake server. This completes the proof of Lemma 13.1.  $\square$

### 13.3 Lower Bound of $\text{IPmod}3_n$

Using the above lemma, we prove the following lemma which extends the theorem of Linial and Shraibman [46] from the two-party model to the server model. Our proof makes use of XOR games as in [42] (attributed to Buhrman). For any boolean function  $f : \mathcal{X} \times \mathcal{Y} \rightarrow \{0,1\}$ , let  $A_f$  be a  $|\mathcal{X}|$ -by- $|\mathcal{Y}|$  matrix such that  $A_f[x, y] = (-1)^{f(x,y)}$ . Recall that for any matrix  $A$ ,  $\|A\|_1 = \sum_{i,j} |A_{i,j}|$ .

**Lemma 13.2.** *For boolean function  $f$  and  $0 \leq \epsilon < 1/4$*

$$4^{2Q_\epsilon^{*,\text{server}}(f)} \geq \max_M \frac{\langle A_f, M \rangle - 2\epsilon \|M\|_1}{\gamma_2^*(M)} = \gamma_2^{2\epsilon}(A_f).$$

*Proof.* We first prove the following claim.

**Claim 13.3.** *For any boolean functions  $f, g$  on the same domain, probability distribution  $\pi$  and  $0 \leq \epsilon \leq 1$ ,*

$$\text{Bias}_\pi(g) \geq \frac{\langle A_f, A_g \circ \pi \rangle - 2\epsilon}{4^{2Q_\epsilon^{*,\text{server}}(f)}}.$$

*Proof.* First, suppose that when receive input  $(x, y)$ , Alice and Bob can somehow compute  $f(x, y)$  and use this as an answer to the XOR game (e.g., Alice and Bob returns  $f(x, y)$  and 1 to the referee respectively). What is the bias this strategy can achieve? Since the probability of winning is  $\sum_{x,y: f(x,y)=g(x,y)} \pi(x, y)$ , the bias is straightforwardly

$$\sum_{\substack{x,y \\ f(x,y)=g(x,y)}} \pi(x, y) - \sum_{\substack{(x,y) \\ f(x,y) \neq g(x,y)}} \pi(x, y) = \sum_{x,y} \pi(x, y) A_f[x, y] A_g[x, y] = \langle A_f, A_g \circ \pi \rangle$$

Let  $\mathcal{A}$  be an  $\epsilon$ -error protocol for computing  $f$  in the server model and  $\mathcal{A}(x, y)$  be the output of  $\mathcal{A}$  (which could be randomized) on input  $(x, y)$ . Now suppose that Alice and Bob use  $\mathcal{A}(x, y)$  to play the XOR game. Then the winning probability will decrease by at most  $\epsilon$ . Thus the bias is at least

$$\langle A_f, A_g \circ \pi \rangle - 2\epsilon. \tag{11}$$

Now suppose that Alice and Bob use protocol  $\mathcal{A}'$  from Lemma 13.1 with  $\epsilon_0 = \epsilon_1 = \epsilon$  to play the XOR game. With probability  $1 - 4^{-2Q_\epsilon^{*,\text{server}}(f)}$ ,  $\mathcal{A}'$  will output randomly; this means that the bias is 0. Otherwise,  $\mathcal{A}'$  will behave as an  $\epsilon$ -error algorithm. Thus, we conclude from Eq.(11) that the bias is at least

$$4^{-2Q_\epsilon^{*,\text{server}}(f)} (\langle A_f, A_g \circ \pi \rangle - 2\epsilon).$$

This completes the claim.  $\square$

Thus, for any  $\pi$

$$4^{2Q_\epsilon^{*,server}(f)} \geq \frac{\langle A_f, A_g \circ \pi \rangle - 2\epsilon}{\text{Bias}_\pi(g)}.$$

Note that  $\text{Bias}_\pi(g) = \gamma_2^*(A_g \circ \pi)$  [66] (also see [42, Theorem 5.2]). So,

$$4^{2Q_\epsilon^{*,server}(f)} \geq \frac{\langle A_f, A_g \circ \pi \rangle - 2\epsilon}{\gamma_2^*(A_g \circ \pi)}.$$

Since this is true for any  $\pi$  and  $g$ ,

$$4^{2Q_\epsilon^{*,server}(f)} \geq \max_{\pi, g} \frac{\langle A_f, A_g \circ \pi \rangle - 2\epsilon}{\gamma_2^*(A_g \circ \pi)} = \max_M \frac{\langle A_f, M \rangle - 2\epsilon \|M\|_1}{\gamma_2^*(M)}.$$

This proves the first inequality in Lemma 13.2.

For the second inequality, we use Proposition 1 in [43] (proved in [42]) which states that for any norm  $\Phi$ , matrix  $A$  and  $0 \leq \alpha < 1$ , the  $\alpha$ -approximate norm is

$$\Phi^\alpha(A) = \max_W \frac{|\langle A, W \rangle| - \alpha \|W\|_1}{\Phi^*(W)}.$$

This means that  $\gamma_2^{2\epsilon}(A_f) = \max_M \frac{|\langle A_f, M \rangle| - 2\epsilon \|M\|_1}{\gamma_2^*(M)}$  as claimed.  $\square$

For finite sets  $X, Y$ , and  $E$ , a function  $f : E^n \rightarrow \{0, 1\}$ , and a function  $g : X \times Y \rightarrow E$ , the *block composition* of  $f$  and  $g$  is the function  $f \circ g^n : X^n \times Y^n \rightarrow \{0, 1\}$  defined by  $(f \circ g^n)(x, y) = f(g(x^1, y^1), \dots, g(x^n, y^n))$  where  $(x^i, y^i) \in X \times Y$  for all  $i = 1, \dots, n$ . For any boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ , let  $f'$  be such that, for all  $x \in \{0, 1\}^n$ ,  $f'(x) = -1$  if  $f(x) = 0$  and  $f'(x) = 1$  otherwise. The  $\epsilon$ -approximate degree of  $f$ , denoted by  $\deg_\epsilon(f)$  is the least degree of a real polynomial  $p$  such that  $|f'(x) - p(x)| \leq \epsilon$  for all  $x \in \{0, 1\}^n$ . We say that  $g$  is *strongly balanced* if all rows and columns in the matrix  $A_g$  sum to zero. For any  $m$ -by- $n$  matrix  $A$ , let  $\text{size}(A) = m \times n$ . We now prove a “server-model version” of Lee and Zhang’s theorem [43, Theorem 8]. Our proof is essentially the same as their proof (also see [42, Theorem 7.6]).

**Lemma 13.4.** *For any finite sets  $X, Y$ , let  $g : X \times Y \rightarrow \{0, 1\}$  be any strongly balanced function. Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be an arbitrary function. Then*

$$Q_\epsilon^{*,server}(f \circ g^n) \geq \deg_{4\epsilon}(f) \log_2 \left( \frac{\sqrt{|X||Y|}}{\|A_g\|} \right) - O(1)$$

for any  $0 < \epsilon < 1/4$ .

*Proof.* We simply follow the proof of Lee and Zhang [43] and use Lemma 13.2 instead of Linial-Shraibman’s theorem. First, we note the following inequality which follows from the definition of  $\gamma_2$ : For any  $\delta \geq 0$  and  $m$ -by- $n$  matrix  $A$ ,

$$\gamma_2^\delta(A) = \min_{B: \|B-A\|_\infty \leq \delta} \gamma_2(B) \geq \min_{B: \|B-A\|_\infty \leq \delta} \frac{\|B\|_{tr}}{\sqrt{\text{size}(B)}} = \frac{\|A\|_{tr}^\delta}{\sqrt{\text{size}(A)}}$$

where the first and last equalities are by definition of the approximate norm (see, e.g., [43, Definition 4]) and the inequality is by the definition of  $\gamma_2$  norm (see, e.g., [43, Definition 1]). Using  $A = A_{f \circ g}$  which is an  $|X|$ -by- $|Y|$  matrix, we have

$$\gamma_2^\delta(A_{f \circ g}) \geq \frac{\|A_{f \circ g}\|_{tr}^\delta}{\sqrt{\text{size}(A_{f \circ g})}}. \quad (12)$$

The following claim is shown in the proof of Theorem 8 in [43].



**Claim 13.5** ([43]).

$$\frac{\|A_{f \circ g}\|_{tr}^\delta}{\sqrt{\text{size}(A_{f \circ g})}} \geq \delta \left( \frac{\sqrt{|X||Y|}}{\|A_g\|} \right)^{\deg_{2\delta}(f)}. \quad (13)$$

*Proof.* We note the following lemma (noted as Lemma 1 in [43]) which shows that there exists a *dual polynomial* of  $f$  which is a polynomial  $v$  which certifies that the approximate polynomial degree of  $f$  is at least a certain value.

**Lemma 13.6** ([60, 61]). *For any  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ , let  $f'$  be such that  $f'(z) = (-1)^{f(z)}$  and  $d = \deg_\delta(f)$ . Then, there exists a function  $v : \{0, 1\}^n \rightarrow \mathbb{R}$  such that*

1.  $\langle v, \chi_T \rangle = 0$  for every character  $\chi_T$  with  $|T| < d$ .
2.  $\|v\|_1 = 1$ .
3.  $\langle v, f' \rangle \geq \delta$ .

Let  $v$  be a dual polynomial of  $f$  as in the above lemma. We will use  $B = (\frac{2^n}{\text{size}(A_g)})A_{v \circ g}$  as a “witness matrix”, i.e.,

$$B[x, y] = \frac{2^n}{\text{size}(A_g)^n} v(g(x_1, y_1), \dots, g(x^n, y^n)). \quad (14)$$

It follows that

$$\langle A_{f \circ g}, B \rangle = \frac{2^n}{\text{size}(A_g)^n} \langle M_{f \circ g}, A_{v \circ g} \rangle \quad (15)$$

$$= \frac{2^n}{\text{size}(A_g)^n} \sum_{x, y} f(g(x^1, y^1), \dots, g(x^n, y^n)) v(g(x^1, y^1), \dots, g(x^n, y^n)) \quad (16)$$

$$= \frac{2^n}{\text{size}(A_g)^n} \sum_{z \in \{0, 1\}^n} \left( f(z) v(z) \left( \sum_{\substack{x, y: \\ g(x^i, y^i) = z_i \\ \forall 1 \leq i \leq n}} 1 \right) \right) \quad (17)$$

$$= \frac{2^n}{\text{size}(A_g)^n} \sum_{z \in \{0, 1\}^n} \left( f(z) v(z) \prod_{i=1}^n \left( \sum_{\substack{x^i, y^i: \\ g(x^i, y^i) = z_i}} 1 \right) \right) \quad (18)$$

$$= \frac{2^n}{\text{size}(A_g)^n} \sum_{z \in \{0, 1\}^n} \left( f(z) v(z) \left( \sum_{\substack{x', y': \\ g(x', y') = z_i}} 1 \right)^n \right) \quad (19)$$

$$= \sum_{z \in \{0, 1\}^n} f(z) v(z) \quad (20)$$

$$= \langle f, v \rangle \quad (21)$$

$$\geq \delta \quad (22)$$

where Eq.(20) is because  $g$  is strongly balanced which implies that  $g$  is balanced, i.e.  $g(x^i, y^i)$  is 0 (and 1) for half of its possible inputs (i.e.  $\text{size}(A_g)/2$  entries of  $A_g$  are 1 (and  $-1$ )); thus,

$$\sum_{\substack{x', y': \\ g(x', y') = z_i}} 1 = \text{size}(A_g)/2.$$

A similar argument and the fact that  $\|v\|_1 = 1$  can be used to show that

$$\|B\|_1 = 1. \quad (23)$$

Now we turn to evaluate the spectral norm  $\|B\|$ . As shown in [43], the strongly balanced property of  $g$  implies that the matrices  $\chi_T \circ g^n$  and  $\chi_S \circ g^n$  are orthogonal for distinct sets  $S, T \subseteq \{0, 1\}^n$ . Note the following fact (Fact 1 in [43]): For any matrices  $A'$  and  $B'$  of the same dimension, if  $A'(B')^\dagger = (A')^\dagger B' = 0$  then  $\|A + B\| = \max\{\|A\|, \|B\|\}$ . Using this fact, we have

$$\|B\| = \frac{2^n}{\text{size}(A_g)^n} \left\| \sum_{T \subseteq [n]} \hat{v}_T A_{\chi_T \circ g^n} \right\| \quad (24)$$

$$= \frac{2^n}{\text{size}(A_g)^n} \max_T |\hat{v}_T| \|\hat{v}_T A_{\chi_T \circ g^n}\| \quad (\text{by the fact above}) \quad (25)$$

$$= \max_T 2^n |\hat{v}_T| \prod_i \frac{\|A_G^{T[i]}\|}{\text{size}(A_g)} \quad (26)$$

$$\leq \max_{T: \hat{v}^T \neq 0} \prod_i \frac{\|A_G^{T[i]}\|}{\text{size}(A_g)} \quad (27)$$

$$= \left( \frac{\|A_g\|}{\sqrt{\text{size}(A_g)}} \right)^d \left( \frac{1}{\text{size}(A_g)} \right)^{n/2} \quad (28)$$

where Eq.(27) is because  $|\hat{v}_T| \leq 1/2^n$  as  $\|v\|_1 = 1$  and Eq.(28) is because  $\|J\| = \sqrt{\text{size}(A_g)}$ .

We note that for any  $0 \leq \epsilon < 1$ , norm  $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}$  and vector  $v \in \mathbb{R}^n$ , the approximate norm is  $\Phi^\epsilon(v) = \max_u \frac{|\langle v, u \rangle| - \epsilon \|u\|_1}{\Phi^*(u)}$  (see, e.g., [42] and [43, Proposition 1]). Note also that if  $\Phi$  is the trace norm then its dual  $\Phi^*$  is the spectral norm (this is noted in [43]). Thus,

$$\|A_{f \circ g^n}\|_{tr}^{\delta/2} = \max_{B'} \frac{|\langle A_{f \circ g^n}, B' \rangle| - (\delta/2) \|B'\|_1}{\|B'\|} \quad (29)$$

$$\geq \frac{|\langle A_{f \circ g^n}, B \rangle| - \delta/2}{\|B\|} \quad (\text{by Eq.(23)}) \quad (30)$$

$$\geq \frac{\delta - \delta/2}{\|B\|} \quad (\text{by Eq.(22)}) \quad (31)$$

$$\geq (\delta/2) \left( \frac{\sqrt{\text{size}(A_g)}}{\|A_g\|} \right)^d (\text{size}(A_g))^{n/2} \quad (\text{by Eq.(28)}) \quad (32)$$

$$\geq (\delta/2) \left( \frac{\sqrt{\text{size}(A_g)}}{\|A_g\|} \right)^d \left( \sqrt{\text{size}(A_{f \circ g})} \right) \quad (33)$$

where the last inequality is because  $\text{size}(A_{f \circ g}) = \text{size}(A_g)^n$ . This completes the proof of the claim.  $\square$

The lemma follows immediately from Eq.(12) and Eq.(13) by plugging in Lemma 13.2:

$$4^{2Q_{\epsilon}^{*,server}(f \circ g^n)} \geq \gamma_2^{2\epsilon}(A_{f \circ g^n}) \geq \frac{\|A_{f \circ g^n}\|_{tr}^{2\epsilon}}{\sqrt{\text{size}(A_{f \circ g^n})}} \geq (2\epsilon) \left( \frac{\sqrt{|X||Y|}}{\|A_g\|} \right)^{\deg_{4\epsilon}(f)}.$$

Lemma 13.4 follows (the term  $2\epsilon$  will contribute to the term “ $-O(1)$ ”).  $\square$

Now, we prove the lower bound of  $\text{IPmod3}_n$ . Our proof essentially follows Sherstov’s proof [60] (also see [42, Section 7.2.3]). We can assume w.l.o.g. that  $n$  is divisible by 4. Consider the *promise version* of  $\text{IPmod3}_n$  where any  $n$ -bit string input  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$  has the property that for any  $0 \leq i \leq (n/4) - 1$ ,

$$\begin{aligned} x_{4i+1}x_{4i+2}x_{4i+3}x_{4i+4} &\in \{0011, 0101, 1100, 1010\} \quad \text{and} \\ y_{4i+1}y_{4i+2}y_{4i+3}y_{4i+4} &\in \{0001, 0010, 1000, 0100\}. \end{aligned}$$

Now we show that the claimed lower bound holds even in this case. This lower bound clearly implies the lower bound of the more general case of  $\text{IPmod3}_n$  where no restriction is put on the input.

Observe that, for any  $(x, y) \in \mathcal{X} \times \mathcal{Y}$ , the function  $\text{IPmod3}$  can be written as

$$f \circ g^{n/4}(x, y) = f(g(x_1 \dots x_4, y_1 \dots y_4), g(x_5 \dots x_8, y_5 \dots y_8), \dots, g(x_{n-3} \dots x_n, y_{n-3} \dots y_n))$$

where

$$g(x_{4i+1} \dots x_{4i+4}, y_{4i+1} \dots y_{4i+4}) = (x_{4i+1} \wedge y_{4i+1}) \vee (x_{4i+2} \wedge y_{4i+2}) \vee (x_{4i+3} \wedge y_{4i+3}) \vee (x_{4i+4} \wedge y_{4i+4})$$

for all  $0 \leq i \leq (n/4) - 1$ , and  $f(z_1, \dots, z_{n/4}) = 1$  if  $z_1 + \dots + z_{n/4}$  can be divided by 3 and 0 otherwise. Note that  $\text{IPmod3}(x, y) = f \circ g^{n/4}(x, y)$  since the promise implies that  $g(x_{4i+1} \dots x_{4i+4}, y_{4i+1} \dots y_{4i+4}) = 1$  if and only if  $x_{4i+1}y_{4i+1} + \dots + x_{4i+4}y_{4i+4} = 1$ . The matrix  $A_g$  is

$$A_g = \begin{matrix} & \begin{matrix} 0001 & 0010 & 1000 & 0100 \end{matrix} \\ \begin{matrix} 0011 \\ 0101 \\ 1100 \\ 1010 \end{matrix} & \begin{pmatrix} -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \end{matrix}$$

which is clearly strongly balanced. It can be checked that this matrix has spectral norm  $\|A_g\| = 2\sqrt{2}$  (see, e.g., [42, Section 7.2.3]). Moreover, by Paturi [53] (see also [16] and [60, Theorem 2.6]),  $\deg_{1/3}(f) = \Theta(n)$ . Thus, Lemma 13.4 implies that

$$\begin{aligned} Q_{1/12}^{*,server}(f \circ g^n) &\geq \deg_{1/3}(f) \log_2 \left( \frac{\sqrt{4 \times 4}}{\|A_g\|} \right) - O(1) \\ &= \deg_{1/3}(f) \log_2 \sqrt{2} - O(1) \\ &= \Omega(n). \end{aligned}$$

We note that the same technique can be used to prove many bounds in the server model similar to bounds in [58, 60, 43].

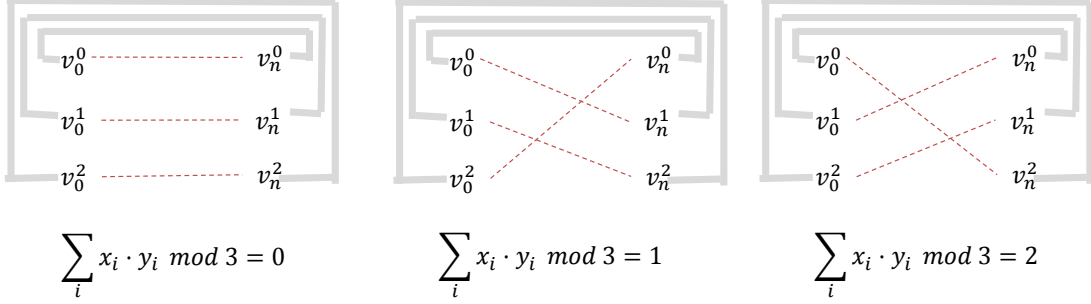


Figure 12: The resulted graph  $G$  in three situations depending on the value of  $\sum_{1 \leq i \leq n} x_i \cdot y_i \bmod 3$ . Dashed lines (in red) represent paths connecting  $v_0^0, \dots, v_0^2$  and  $v_n^0, \dots, v_n^2$ . Thick lines (in gray) show the fact that we identify nodes on two sides, i.e.  $v_0^j = v_n^j$  for all  $0 \leq j \leq 2$ . Our main observation is that  $G$  is a Hamiltonian cycle if and only if  $\sum_{1 \leq i \leq n} x_i \cdot y_i \bmod 3 \neq 0$  (cf. Lemma 14.3).

## 14 Detail of Section 9

First, let us recall that Alice and Bob construct a gadget  $G_i$  using  $x_i$  and  $y_i$  as shown in Fig. 4. Fig. 5 shows how  $G_i$  looks like for each possible value of  $x_i$  and  $y_i$ . It follows immediately that  $G_i$  always consist of three paths which connect  $v_{i-1}^j$  to  $v_i^{(j+x_i \cdot y_i) \bmod 3}$ , as in the following observation.

**Observation 14.1** (Observation 9.2 restated). *For any value of  $(x_i, y_i)$ ,  $G_i$  consists of three paths where  $v_{i-1}^j$  is connected by a path to  $v_i^{(j+x_i \cdot y_i) \bmod 3}$ , for any  $0 \leq j \leq 2$ . Moreover, Alice's (respectively Bob's) edges, i.e. thin (red) lines (respectively thick (blue) lines) in Fig. 4, form a matching that covers all nodes except  $v_i^j$  (respectively  $v_{i-1}^j$ ) for all  $0 \leq j \leq 2$ .*

Finally, we connect gadgets  $G_i$  and  $G_{i+1}$  together by identifying rightmost nodes of  $G_i$  with leftmost nodes of  $G_{i+1}$ , as shown in Fig. 6 (gray lines represent the fact that we identify rightmost nodes of  $G_n$  to leftmost nodes of  $G_1$ ).

**Lemma 14.2** (Lemma 9.3 restated).  *$G$  consists of three paths  $P^0, P^1$  and  $P^2$  where for any  $0 \leq j \leq 2$ ,  $P^j$  has  $v_0^j$  as one end vertex and  $v_n^{(j+\sum_{1 \leq i \leq n} x_i \cdot y_i) \bmod 3}$  as the other.*

*Proof.* We will show that for any  $2 \leq k \leq n$  and  $0 \leq j \leq 2$ ,  $P^j$  has  $v_0^j$  as one end vertex and  $v_k^{(j+\sum_{1 \leq i \leq k} x_i \cdot y_i) \bmod 3}$  as the other. We prove this by induction on  $k$ . Our claim clearly holds for  $k = 2$  by Observation 14.1. Now assume that this claim is true for any  $2 \leq k \leq n - 1$ , i.e.,  $v_0^j$  is connected by a path to  $v_k^{j'}$  where  $j' = (j + \sum_{1 \leq i \leq k} x_i \cdot y_i) \bmod 3$ . By Observation 14.1,  $v_k^{j'}$  is connected by a path to  $v_k^{j''}$  where  $j'' = (j' + x_{k+1} \cdot y_{k+1}) \bmod 3 = (j + \sum_{1 \leq i \leq k+1} x_i \cdot y_i) \bmod 3$  as claimed.  $\square$

**Lemma 14.3.** *Each player's edges form a perfect matching in  $G$ . Moreover,  $G$  is a Hamiltonian cycle if and only if  $\sum_{1 \leq i \leq n} x_i \cdot y_i \bmod 3 \neq 0$ .*

*Proof.* We consider three different values of  $z = \sum_{1 \leq i \leq n} x_i \cdot y_i \bmod 3$  as shown in Fig. 12. If  $z = 0$  then Lemma 14.2 implies that  $v_0^j$  will be connected to  $v_n^j$  by a path, for all  $j$ . After we identify  $v_0^j$  with  $v_n^j$  we will have three distinct cycles, each containing a distinct  $v_0^j = v_n^j$ . If  $z = 1$  then Lemma 14.2 implies that  $v_0^j$  will be connected to  $v_n^{(j+1) \bmod 3}$  by a path. After we identify  $v_0^j$  with  $v_n^j$  we will have one cycle that connects  $v_0^0 = v_n^0$  to  $v_0^1 = v_n^1$  then to  $v_n^2 = v_0^2$ . Similarly, if  $z = 2$  then Lemma 14.2 implies that  $v_0^j$  will be connected to  $v_n^{(j+2) \bmod 3}$  by a path. After we identify  $v_0^j$  with  $v_n^j$  we will have one cycle that connects  $v_0^0 = v_n^0$  to  $v_n^2 = v_0^2$  then to  $v_n^1 = v_0^1$ .  $\square$

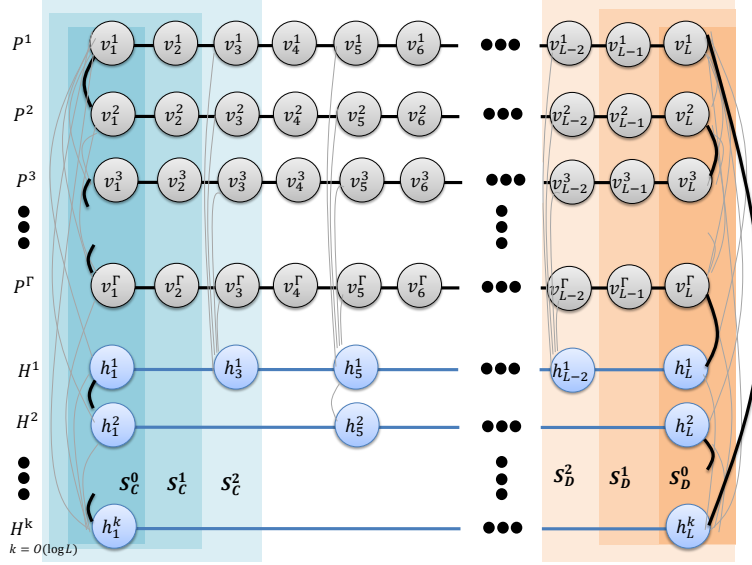


Figure 13: (Fig. 10 reproduced) The network  $N$  which consists of network  $N'$  and some “highways” which are paths with nodes  $h_j^i$  (i.e., nodes in blue). Bold edges show an example of subnetwork  $M$  when the input perfect matchings are  $E_C = \{(u_1, u_2), (u_3, u_4), \dots, (u_{\Gamma+k-1}, u_{\Gamma+k})\}$  and  $E_D = \{(u_2, u_3), (u_4, u_5), \dots, (u_{\Gamma+k}, u_1)\}$ . Pale edges are those in  $N$  but not in  $M$ .

## 15 Full proof of Theorem 10.1

**Theorem 15.1** (Theorem 10.1 restated). *For any  $B, L, \Gamma \geq \log L, \beta \geq 0$  and  $\epsilon_0, \epsilon_1 > 0$ , there exists a  $B$ -model quantum network  $N$  of diameter  $\Theta(\log L)$  and  $\Theta(\Gamma L)$  nodes such that*

$$\text{if } Q_{\epsilon_0, \epsilon_1}^{*, N}(\mathcal{P}(N)) \leq \frac{L}{2} - 2 \text{ then } Q_{\epsilon_0, \epsilon_1}^{*, \text{server}}(\mathcal{P}_\Gamma) = O((B \log L) Q_{\epsilon_0, \epsilon_1}^{*, N}(\mathcal{P}(N)))$$

where  $\mathcal{P}$  can be replaced by Ham and  $(\beta\Gamma)$ -Conn.

### 15.1 Description of the network $N$

In this section we describe the network  $N$  as shown in Fig. 13. We assume that  $L = 2^i + 1$  for some  $i$ . This can be assumed without changing the theorem statement by simply increasing  $L$  to the nearest number of this form.

The two basic units in the construction are *paths* and *highways*. There are  $\Gamma$  paths, denoted by  $P^1, P^2, \dots, P^\Gamma$ , each having  $L$  nodes, i.e., for  $j = 1, 2, \dots, \Gamma$ ,

$$V(P^i) = \{v_1^i, \dots, v_L^i\} \quad \text{and} \quad E(P^i) = \{(v_j^i, v_{j+1}^i) \mid 1 \leq j \leq L-1\}.$$

We construct  $k = \log_2(L-1)$  highways, denoted by  $H^1, \dots, H^k$  where  $H^i$  has the following nodes and edges.

$$V(H^i) = \{h_{1+j2^i}^i \mid 0 \leq j \leq \frac{L-1}{2^i}\} \quad \text{and} \\ E(H^i) = \{(h_{1+j2^i}^i, h_{1+(j+1)2^i}^i) \mid 0 \leq j \leq \frac{L-1}{2^i}\}.$$

For any node  $h_j^1$  we add an edge  $(h_j^1, v_j^1)$  for any  $j$ . Moreover for any node  $h_j^i$  we add an edge  $(h_j^{i-1}, h_j^i)$ . Figure 13 depicts this network. We note the following simple observation.

**Observation 15.2.** *The number of nodes in  $N$  is  $n = \Theta(L\Gamma)$  and its diameter is  $\Theta(\log L)$ .*

## 15.2 Simulation

For any  $0 \leq t \leq (L/2) - 2$ , we partition  $V(N)$  into three sets, denoted by  $S_C^t$ ,  $S_D^t$  and  $S_S^t$ , as follows (also see Fig. 13).

$$S_C^t = \{v_j^i, h_j^i \mid 1 \leq i \leq \Gamma, 1 \leq j \leq t+1\}, \quad (34)$$

$$S_D^t = \{v_j^i, h_j^i \mid 1 \leq i \leq \Gamma, L-t \leq j \leq L\}, \quad (35)$$

$$S_S^t = V(N) \setminus (S_C^t \cup S_D^t). \quad (36)$$

Let  $\mathcal{A}$  be any quantum distributed algorithm on network  $N$  for computing a problem  $P$  (which is either **Ham** or  $(\beta\Gamma) - \text{Conn}$ ). Let  $T_{\mathcal{A}}$  be the worst case running time of algorithm  $\mathcal{A}$  (over all inputs). We note that  $T_{\mathcal{A}} \leq (L/2) - 2$ , as assumed in the theorem statement. We show that Carol, David and the server can solve problem  $P$  on  $(\Gamma + k)$ -node input graph using small communication, essentially by “simulating”  $\mathcal{A}$  on some input subnetwork  $M$  corresponding to  $G = (U, E_C \cup E_D)$  in the following sense. When receiving  $E_C$  and  $E_D$ , the three parties will construct a subnetwork  $M$  of  $N$  (without communication) in such a way that  $M$  is a 1-input of problem  $P$  (e.g.,  $M$  is a Hamiltonian cycle) if and only if  $G = (U, E_C \cup E_D)$  is. Next, they will simulate algorithm  $\mathcal{A}$  in such a way that, at any time  $t$  and for each node  $v_j^i$  in  $N$ , there will be exactly one party among Carol, David and the server that knows *all information that  $v_j^i$  should know in order to run algorithm  $\mathcal{A}$* , i.e., the (quantum) state of  $v_j^i$  as well as the messages (each consisting of  $B$  quantum bits) sent to  $v_j^i$  from its neighbors at time  $t$ . The party that knows this information will pretend to be  $v_j^i$  and apply algorithm  $\mathcal{A}$  to get the state of  $v_j^i$  at time  $t+1$  as well as the messages that  $v_j^i$  will send to its neighbors at time  $t+1$ . We say that this party *owns*  $v_j^i$  at time  $t$ . Details are as follows.

We will define a server-model protocol  $\mathcal{A}'$  that guarantees that, at any time  $t$ , Carol, David and the server will own nodes in sets  $S_C^t$ ,  $S_D^t$  and  $S_S^t$ , respectively, at time  $t$ . That is, Carol’s workspace, denoted by  $H_{C,C}$ , contains all qubits in  $H_{vv'}$ , for any  $v \in S_C^t$  and  $v' \in V(N)$ , resulting from  $t$  rounds of an execution of  $\mathcal{A}$ . Similarly, David’s (respectively the server’s) workspace, denoted by  $H_{D,D}$  (respectively  $H_{S,S}$ ), contains all qubits in  $H_{vv'}$  for any  $v \in S_D^t$  (respectively  $v \in S_S^t$ ) and  $v' \in V(N)$  resulting from  $t$  rounds of  $\mathcal{A}$ . In other words, if after  $t$  rounds of  $\mathcal{A}$  network  $N$  has state

$$|\psi_M^t\rangle = \sum_w \left( \alpha_w \bigotimes_{v,v' \in V(N)} |\psi_{w,M}^t(v, v')\rangle \right),$$

then we will make sure that the server model has state

$$|\Psi_G^t\rangle = \sum_w \left( \alpha_w \bigotimes_{i,j \in \{C,D,S\}} |\Psi_{w,G}^t(i, j)\rangle \right)$$

where  $|\Psi_{w,G}^t(i, j)\rangle = |0\rangle$ , for any  $i, j \in \{C, D, S\}$  such that  $i \neq j$ , and for any  $i \in \{C, D, S\}$

$$|\Psi_{w,G}^t(i, i)\rangle = \bigotimes_{v \in S_i^t, v' \in V(N)} |\psi_{w,M}^t(v', v)\rangle. \quad (37)$$

Let  $\Gamma' = \Gamma + k$ . Fix any  $\Gamma'$ -node input graph  $G = (U, E_C \cup E_D)$  of problem  $P$  where  $E_C$  and  $E_D$  are edges given to Carol and David respectively. Let  $U = \{u_1, \dots, u_{\Gamma'}\}$ . For convenience, for any  $1 \leq j \leq k$ , let  $v_1^{\Gamma+j} = h_1^j$  and  $v_L^{\Gamma+j} = h_L^j$ . We construct a subnetwork  $M$  of  $N$  as follows. For any  $i \neq j$ , we mark  $v_1^i v_1^j$  as participating in  $M$  if and only if  $u_i u_j \in E_C$ . Note that this knowledge must be kept in qubits in  $H_{v_1^i v_1^i}$  and  $H_{v_1^j v_1^j}$  in network  $N$  as we require each node to know whether

edges incident to it are in  $M$  or not. This means that this knowledge must be stored in  $H_{C,C}$  since  $v_1^i, v_1^j \in S_C^0$ . This can be guaranteed without any communication since Carol knows  $E_C$ . Similarly, we mark  $v_L^i, v_L^j$  as participating in  $M$  if and only if  $u_i u_j \in E_D$ , and this information can be stored in  $H_{D,D}$  without communication. Finally, we let all edges in all paths and highways be in  $M$ . This information is stored in  $H_{S,S}$ . An example of network  $M$  is shown in Fig. 13. To conclude, if the initial state of  $N$  with this subnetwork  $M$  is

$$|\psi_M^0\rangle = \sum_w \left( \alpha_w \bigotimes_{v,v' \in V(N)} |\psi_{w,M}^0(v, v')\rangle \right).$$

then the server model will start with state  $|\Psi_G^0\rangle = \sum_w \left( \alpha_w \bigotimes_{i,j \in \{C,D,S\}} |\Psi_{w,G}^0(i, j)\rangle \right)$  where  $|\Psi_w^0(i, j)\rangle = |0\rangle$ , for any  $i, j \in \{C, D, S\}$  such that  $i \neq j$ , and for any  $i \in \{C, D, S\}$

$$|\Psi_{w,G}^0(i, i)\rangle = \bigotimes_{v \in S_i^0, v' \in V(N)} |\psi_{w,M}^0(v', v)\rangle.$$

Thus Eq.(37) holds for  $t = 0$ . We note the following simple observation.

**Observation 15.3.**  *$G = (U, E_C \cup E_D)$  is a Hamiltonian cycle if and only if  $M$  is a Hamiltonian cycle.  $G$  is connected if and only if  $M$  is connected, and for any  $\delta$ ,  $G$  is  $\delta$ -far from being connected if and only if  $M$  is  $\delta$ -far from being connected.*

Thus, Carol, David and the server can check whether  $G$  is a Hamiltonian cycle if they can check whether  $M$  is a Hamiltonian cycle. Similarly, they can check if  $G$  is connected or  $(\beta\Gamma)$ -far from being connected by checking  $M$ . So, if Eq.(37) can be maintained until  $\mathcal{A}$  terminates then we are done since each server-model player can pretend to be one of the nodes they own and measure the workspace of such node to get the property of  $M$ .

Now suppose that Carol, David and the server have maintained this guarantee until they have executed  $\mathcal{A}$  for  $t-1$  steps, i.e., player  $i$  owns the nodes in  $S_i^{t-1}$  at time  $t-1$ . They maintain the guarantee at step  $t$  as follows. First, each player simulate the internal computation of  $\mathcal{A}$  on nodes they own. That is, for each node  $v \in V(N)$ , the player  $i$  such that  $v \in S_i^{t-1}$  applies the transformation  $C_{t,v}$  (cf. Section 12.1) on qubits in workspace  $\bigotimes_{v' \in V(N)} H_{v'v}$  which is maintained in  $H_{i,i}$  at time  $t-1$ . This means that if after the internal computation  $N$  has state  $|\psi_M^t\rangle = \sum_w \left( \alpha_w \bigotimes_{v,v' \in V(N)} |\psi_{w,M}^t(v, v')\rangle \right)$  then the server model will have state  $|\Upsilon_G^t\rangle = \sum_w \left( \alpha_w \bigotimes_{i,j \in \{C,D,S\}} |\Upsilon_{w,G}^t(i, j)\rangle \right)$  where  $|\Upsilon_w^t(i, j)\rangle = |0\rangle$ , for any  $i \neq j$ , and  $|\Upsilon_{w,G}^t(i, i)\rangle = \bigotimes_{v \in S_i^t, v' \in V(N)} |\psi_{w,M}^t(v', v)\rangle$  for any  $i$ . Note that the server model players can simulate the internal computation of  $\mathcal{A}$  without any communication since a player that owns node  $v$  has all information needed to simulate an internal computation of  $v$  (i.e., the state of  $v$  as well as all messages  $v$  received at time  $t-1$ ).

At this point, for any  $i \in \{C, D, S\}$ , player  $i$ 's space contains the current state and out-going messages of every node  $v \in S_i^{t-1}$ . They will need to receive some information in order to guarantee that they own nodes in  $S_i^t$ . First, consider Carol. Let  $S'_C$  be the set of rightmost nodes in the set  $S_C^{t-1}$ , i.e.  $S'_C$  consists of  $v_{i+1}^i$  and  $h_j^i$  for all  $i$  and  $j = \arg \max_j \{h_j^i \in S_C^{t-1}\}$ .

Note that Carol already has the workspace and all incoming messages of nodes in  $S_C^{t-1} \setminus S'_C$  at time  $t$ . This is because for any  $v \in S_C^{t-1} \setminus S'_C$ , Carol already has qubits in  $H_{v'v}$  for all  $v' \in V(N)$ . For each  $v \in S'_C$ , Carol is missing the messages sent from  $v$ 's right neighbor; i.e., Carol does not have qubits in  $H_{v_{i+2}^i, v_{i+1}^i}$  and  $H_{h_j^i, h_j^i}$  for all  $i, j = \arg \max_j \{h_j^i \in S_C^{t-1}\}$  and  $j' = \arg \min_{j'} \{h_{j'}^i \notin S_C^{t-1}\}$ . Since  $S'_C \subseteq S_C^t$ , we need to make sure that Carol has all information of nodes in  $S'_C$  at time  $t$ .

For a non-highway node  $v_{t+1}^i$ , for all  $i$ , this can be done by letting the server who owns  $v_{t+2}^i$  send to Carol a message sent from  $v_{t+2}^i$  to  $v_{t+1}^i$  at time  $t$ , i.e., qubits in  $H_{v_{t+2}^i v_{t+1}^i}$ . For highway node  $h_j^i$  for all  $i$  and  $j = \arg \max_j \{h_j^i \in S_C^{t-1}\}$ , its right neighbor  $h_{j'}^i$ , where  $j' = \arg \min_{j'} \{h_{j'}^i \notin S_C^{t-1}\}$ , might be owned by David or the server. In any case, we let the owner of  $h_{j'}^i$  send to Carol the message sent from  $h_{j'}^i$  to  $h_j^i$  at time  $t$ , i.e., qubits in  $H_{h_{j'}^i h_j^i}$ . The cost of doing this is zero if  $h_{j'}^i$  belongs to the server and at most  $B$  if  $h_{j'}^i$  belongs to David since the message size is at most  $B$ . In any case, the total cost will be at most  $Bk$  since there are  $k$  highways. We can thus make sure that Carol gets the information of nodes in  $S_C^{t-1}$  at time  $t$  at the total cost of at most  $Bk$ .

In addition to this, Carol needs to get information of nodes in  $S_C^t \setminus S_C^{t-1}$  at time  $t$ . This means that, for any  $v \in S_C^t \setminus S_C^{t-1}$  she has to receive the qubits stored in  $H_{v'v}$  for all  $v' \in V(N)$ . For any non-highway node  $v_{t+2}^i \in S_C^t \setminus S_C^{t-1}$ , it can be checked from the definition that  $v_{t+2}^i$  and all its neighbors are in  $S_C^{t-1} \cup S_S^{t-1}$ . So, we can make sure that Carol owns  $v_{t+2}^i$  by letting the server send to Carol the workspace of  $v_{t+2}^i$  and messages sent to  $v_{t+2}^i$  by its neighbors in  $S_S^{t-1}$  (i.e. qubits in  $H_{v'v_{t+2}^i}$  for all  $v' \in S_S^{t-1}$ ). This communication is again free. For a highway node  $h_j^i$  in  $S_C^t \setminus S_C^{t-1}$ , it can be checked from the definition that  $h_j^i$  as well as all its *non-highway* neighbors are in  $S_C^{t-1} \cup S_S^{t-1}$ . The only neighbor of  $h_j^i$  that might be in  $S_D^{t-1}$  is its right neighbor, say  $h_{j'}^i$ , in the highway. If  $h_{j'}^i$  is in  $S_D^{t-1}$  then David has to send to Carol the message sent from  $h_{j'}^i$  to  $h_j^i$ . This has cost at most  $B$ . So, Carol can obtain the workspace of  $h_j^i$  as well as all messages sent to  $h_j^i$  at the cost of  $B$ . Since there are  $k$  highway nodes in  $S_C^t \setminus S_C^{t-1}$ , the total cost for Carol to obtain information needed to maintain nodes in  $S_C^t \setminus S_C^{t-1}$  is  $Bk$ . We conclude that Carol can obtain all information needed to own nodes in  $S_C^t$  at time  $t$  at the cost of  $2Bk$ .

We can do the same thing to guarantee that David owns all nodes in  $S_D^t$  at time  $t$  at the cost of  $2Bk$ . Now we make sure that the server own nodes in  $S_S^t$ . First, observe that the server already has the workspace of all nodes in  $S_S^t$  since  $S_S^t \subseteq S_S^{t-1}$ . Moreover, the server already has all messages sent to all non-highway nodes in  $S_S^t$  (i.e.  $v_j^i$  for all  $t+2 \leq j \leq L-t-1$  and  $1 \leq i \leq \Gamma$ ) since all of their neighbors are in  $S_S^{t-1}$ . Additionally, each leftmost highway node  $h_j^i \in S_S^t$ , for any  $i$  and  $j = \arg \min_j \{h_j^i \in S_S^t\}$ , has at most one neighbor in  $S_C^{t-1}$  (i.e., its right neighbor in the highway). Similarly, each rightmost highway node  $h_j^i \in S_S^t$ , for any  $i$  and  $j' = \arg \max_{j'} \{h_{j'}^i \in S_S^t\}$ , has at most one neighbor in  $S_D^{t-1}$  (i.e., its right neighbor in the highway). Thus, the server needs to obtain from Carol and David at most  $2B$  qubits to maintain  $h_j^i$  and  $h_{j'}^i$ . Since there are  $k$  highways, the server needs at most  $2kB$  qubits total from Carol and David. We thus conclude that the players can maintain Eq.(37) at the cost of  $6kB = O(B \log L)$  qubits per round as desired.

As noted earlier, the server-model players will simulate  $\mathcal{A}$  until  $\mathcal{A}$  terminates. Then they can measure the workspace of nodes they own to check whether  $M$  is a 0- or 1-input of problem  $P$ . Observation 15.3 implies that they can use this answer to answer whether  $G$  is a 0- or 1-input with the same error probability. Since each round of simulation requires a communication complexity of  $O(B \log L)$  and the simulation is done for  $T_{\mathcal{A}} \leq Q_{\epsilon_0, \epsilon_1}^{*,N}(P(N))$  rounds, the total communication complexity is  $O((B \log L)Q_{\epsilon_0, \epsilon_1}^{*,N}(P(N)))$  as claimed.



## References

- [1] Scott Aaronson and Andris Ambainis. Quantum search of spatial regions. *Theory of Computing*, 1(1):47–79, 2005. Also in FOCS’03.
- [2] László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *FOCS*, pages 337–347, 1986.
- [3] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004. Also in FOCS’02.
- [4] J S Bell. On the einstein-podolsky-rosen paradox. *Physics*, 1(3):195–200, 1964.
- [5] M. Ben-Or and A. Hassidim. Fast quantum byzantine agreement. In *STOC*, pages 481–485, 2005.
- [6] Jop Briët. *Grothendieck Inequalities, Nonlocal Games and Optimization*. PhD thesis, Universiteit van Amsterdam, 2011.
- [7] Anne Broadbent and Alain Tapp. Can quantum mechanics help distributed computing? *SIGACT News*, 39(3):67–76, 2008.
- [8] H. Buhrman and H. Rohrig. Distributed quantum computing. In *Proceedings of Mathematical Foundations of Computer Science (MFCS), LNCS 2747*, pages 1–20, 2003.
- [9] H. Buhrman, W. van Dam, P. Hoyer, and A. Tapp. Quantum multiparty communication complexity. *Physical Review A*, 60:2737–2741, 1999.
- [10] Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. classical communication and computation. In *STOC*, pages 63–68, 1998.
- [11] Bogdan S. Chlebus, Dariusz R. Kowalski, and Michal Strojnowski. Scalable quantum consensus for crash failures. In *DISC*, pages 236–250, 2010.
- [12] R. Cleve and H. Buhrman. Substituting quantum entanglement for communication. *Physical Review A*, 56(2):1201–1204, 1997.
- [13] Edith Cohen. Size-Estimation Framework with Applications to Transitive Closure and Reachability. *J. Comput. Syst. Sci.*, 55(3):441–453, 1997. Also in FOCS’94.
- [14] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. In *STOC*, pages 363–372, 2011. Available at <http://arxiv.org/abs/1011.3049>.
- [15] Ronald de Wolf. Quantum communication and complexity. *Theoretical Computer Science*, 287(1):337–352, 2002.
- [16] Ronald de Wolf. A note on quantum algorithms and the minimal degree of  $\epsilon$ -error polynomials for symmetric functions. *Quantum Information & Computation*, 8(10):943–950, 2010.
- [17] Vasil S. Denchev and Gopal Pandurangan. Distributed quantum computing: a new frontier in distributed systems or science fiction? *SIGACT News*, 39(3):77–95, 2008.
- [18] Devdatt P. Dubhashi, Fabrizio Grandioni, and Alessandro Panconesi. Distributed Algorithms via LP Duality and Randomization. In *Handbook of Approximation Algorithms and Metaheuristics*. Chapman and Hall/CRC, 2007.
- [19] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47(10):777–780, May 1935.
- [20] Michael Elkin. Distributed approximation: a survey. *SIGACT News*, 35(4):40–57, 2004.
- [21] Michael Elkin. Computing almost shortest paths. *ACM Transactions on Algorithms*, 1(2):283–323, 2005. Also in PODC’01.
- [22] Michael Elkin. An Unconditional Lower Bound on the Time-Approximation Trade-off for the Distributed Minimum Spanning Tree Problem. *SIAM J. Comput.*, 36(2):433–456, 2006. Also in STOC’04.
- [23] Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. Networks cannot compute their diameter in sublinear time. In *SODA*, pages 1150–1162, 2012.
- [24] Sascha Gaertner, Mohamed Bourennane, Christian Kurtsiefer, Adan Cabello, and Harald Weinfurter. Experimental demonstration of a quantum protocol for byzantine agreement and liar detection. *PHYS.REV.LETT.*, 100:070504, 2008.
- [25] J. Garay, S. Kutten, and D. Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM J. on Computing*, 27:302–316, 1998. Also in FOCS’93.
- [26] Cyril Gavaille, Adrian Kosowski, and Marcin Markiewicz. What can be observed locally? In *DISC*, pages 243–257, 2009.

- [27] A. S. Holevo. Bounds for the quantity of information transmitted by a quantum communication channel. *Problemy Peredachi Informatsii*, 9(3):3–11, 1973. English translation in *Problems of Information Transmission*, 9:177–183, 1973.
- [28] Gabor Ivanyos, Hartmut Klauck, Troy Lee, Miklos Santha, and Ronald de Wolf. New bounds on the classical and quantum communication complexity of some graph properties. *Manuscript*, 2012.
- [29] Bala Kalyanasundaram and Georg Schnitger. The Probabilistic Communication Complexity of Set Intersection. *SIAM J. Discrete Math.*, 5(4):545–557, 1992.
- [30] Maleq Khan, Fabian Kuhn, Dahlia Malkhi, Gopal Pandurangan, and Kunal Talwar. Efficient distributed approximation algorithms via probabilistic tree embeddings. In *PODC*, pages 263–272, 2008.
- [31] Maleq Khan and Gopal Pandurangan. A fast distributed approximation algorithm for minimum spanning trees. *Distributed Computing*, 20(6):391–402, 2008. Also in DISC’06.
- [32] Hartmut Klauck and Ronald de Wolf. Fooling one-sided quantum protocols. *Manuscript*, 2012.
- [33] Hirotada Kobayashi, Keiji Matsumoto, and Seiichiro Tani. Brief announcement: exactly electing a unique leader is not harder than computing symmetric functions on anonymous quantum networks. In *PODC*, pages 334–335, 2009.
- [34] Hirotada Kobayashi, Keiji Matsumoto, and Seiichiro Tani. Computing on anonymous quantum network. *CoRR*, abs/1001.5307, 2010.
- [35] Liah Kor, Amos Korman, and David Peleg. Tight bounds for distributed mst verification. In *STACS*, pages 69–80, 2011.
- [36] F. Kuhn, T. Moscibroda, and R. Wattenhofer. The Price of Being Near-Sighted. In *17th SODA*, 2006.
- [37] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In *PODC*, pages 300–309, 2004.
- [38] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *CoRR*, abs/1011.5470, 2010.
- [39] Fabian Kuhn and Rotem Oshman. The complexity of data aggregation in directed networks. In *DISC*, pages 416–431, 2011.
- [40] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, New York, NY, USA, 1997.
- [41] Shay Kutten and David Peleg. Fast Distributed Construction of Small  $k$ -Dominating Sets and Applications. *J. Algorithms*, 28(1):40–66, 1998. Also in PODC’95.
- [42] Troy Lee and Adi Shraibman. Lower bounds in communication complexity. *Foundations and Trends in Theoretical Computer Science*, 3(4):263–398, 2009.
- [43] Troy Lee and Shengyu Zhang. Composition theorems in communication complexity. In *ICALP (1)*, pages 475–489, 2010.
- [44] Frank T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, 1 edition, 1991.
- [45] Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.
- [46] Nati Linial and Adi Shraibman. Lower bounds in communication complexity based on factorization norms. *Random Struct. Algorithms*, 34(3):368–394, 2009. Also in STOC’07.
- [47] Zvi Lotker, Boaz Patt-Shamir, and David Peleg. Distributed MST for constant diameter graphs. *Distributed Computing*, 18(6):453–460, 2006. Also in PODC’01.
- [48] Nancy Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
- [49] Danupon Nanongkai, Atish Das Sarma, and Gopal Pandurangan. A tight unconditional lower bound on distributed randomwalk computation. In *PODC*, pages 257–266, 2011.
- [50] Ashwin Nayak. Optimal lower bounds for quantum automata and random access codes. In *FOCS*, pages 369–377, 1999.
- [51] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (Cambridge Series on Information and the Natural Sciences)*. Cambridge University Press, 1 edition, January 2004.
- [52] Sudebkumar Prasant Pal, Sudhir Kumar Singh, and Somesh Kumar. Multi-partite quantum entanglement versus randomization: Fair and unbiased leader election in networks, 2003.
- [53] Ramamohan Paturi. On the degree of polynomials that approximate symmetric boolean functions (preliminary version). In *STOC*, pages 468–474, 1992.
- [54] David Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.
- [55] David Peleg and Vitaly Rubinfeld. A Near-Tight Lower Bound on the Time Complexity of Distributed

- Minimum-Weight Spanning Tree Construction. *SIAM J. Comput.*, 30(5):1427–1442, 2000. Also in FOCS’99.
- [56] R. Raz. Exponential separation of quantum and classical communication complexity. In *STOC*, pages 358–369, 1999.
  - [57] Ran Raz and Boris Spieker. On the ”log rank”-conjecture in communication complexity. *Combinatorica*, 15(4):567–588, 1995. Also in FOCS’93.
  - [58] A A Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya: Mathematics*, 67(1):145, 2003.
  - [59] Alexander A. Razborov. On the Distributional Complexity of Disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992. Also in ICALP’90.
  - [60] Alexander A. Sherstov. The pattern matrix method. *SIAM J. Comput.*, 40(6):1969–2000, 2011. Also in STOC’08.
  - [61] Yaoyun Shi and Yufan Zhu. Quantum communication complexity of block-composed functions. *Quantum Info. Comput.*, 9(5):444–460, May 2009.
  - [62] Jukka Suomela. Survey of local algorithms. *ACM Computing Surveys*, to appear.
  - [63] Amnon Ta-Shma. Classical versus quantum communication complexity. *SIGACT News*, 30(3):25–34, 1999.
  - [64] Seiichiro Tani, Hirotada Kobayashi, and Keiji Matsumoto. Exact quantum algorithms for the leader election problem. In *STACS*, pages 581–592, 2005.
  - [65] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 1994.
  - [66] Boris Tsirelson. Quantum analogues of the bell inequalities: the case of two spatially separated domains. *Journal of Soviet Mathematics*, 36:557–570, 1987.
  - [67] John Watrous. Guest column: an introduction to quantum information and quantum circuits 1. *SIGACT News*, 42(2):52–67, 2011.
  - [68] Bang Ye Wu, Giuseppe Lancia, Vineet Bafna, Kun-Mao Chao, R. Ravi, and Chuan Yi Tang. A polynomial-time approximation scheme for minimum routing cost spanning trees. *SIAM J. Comput.*, 29(3):761–778, 1999. Also in SODA’98.